

ROBOT HANDS - ARDUINO BOT - LEGO BOT

SERVO

FOR THE ROBOT INNOVATOR
www.servomagazine.com

MAGAZINE
January 2011

CIRQUE Du Robot

The SmartSensor Lite
from CATCAN Creative
gets put through
its paces

♦ Arduino-like
results from the
Arduino-compatible
PIC-based
Pinguino

Combat Zone ♦

Identifying unknown
brushed DC motors •

Affordable 2.4 GHz •

Using Li-poly or Li-ion
batteries safely •

Brushed DC Motor
Tutorial •



U.S. \$5.50 CANADA \$7.00



0 71486 02422 4 0.1>

INVEST in the BEST

YOUR ROBOT WILL THANK YOU

At Hitec, we know how much time and money you dedicate to your robot hobby. So why not make sure your servos are delivering what they promise? Our technologically advanced mega servos are tough enough for even your most challenging projects. Designed with our newest high resolution "G2.5" 12-bit programmable digital circuit and indestructible titanium gears, the HS-7980TH and HS-M7990TH give mega torque and speed with pinpoint accuracy. Built to last, these servos bring unprecedented power and sustainability to your investment.



Our HS-7980TH employs durable mechanical potentiometer technology while the HS-M7990TH utilizes the first ever magnetic encoder.



Model	6 Volts		7.4 Volts		Part#	Dimensions	Weight
	Speed	Torque	Speed	Torque			
HS-7980TH	0.20	500 oz.in	0.17	611 oz.in	37980S	1.72 x 0.88 x 1.57 in	2.70 oz
HS-M7990TH	0.20	500 oz.in	0.17	611 oz.in	37990S	1.72 x 0.88 x 1.57 in	2.70 oz



We Serve you Right!

12115 Paine Street, Poway, CA 92064 • 858-748-6948 • www.hitecrd.com

X-TREME geek

x-treme Geek is wired
in to what you want.

You want that Big Reaction when your loved one opens their gift. We guarantee you'll get it when you shop with us. From never before seen modern marvels to kitschy cool blasts from the past, X-treme Geek has the gifts to make this holiday the best one yet. Shop x-tremegeek.com.

2500941 \$16.95
EchoBot Voice Messenger

- Detects movement up to 3 feet away



Computer not included

EchoBot Delivers a Voice Message when Somebody Moves Nearby

Record messages for your friends and coworkers, and the EchoBot will play them back when they approach. Up to ten seconds of audio will play back when someone triggers the motion detector (built into the "eye," of course). Bendable legs and suction cup feet make it easy to position the EchoBot anywhere around your home, office or car. Colors vary. Size: 7" tall.

A Carpentry Kit for a Robot?

NEW

Maybe early robots were made of wood....or maybe it's just an awesome idea who's time is long overdue. This kit is supplied with pre-punched wooden boards, gears, shafts, switch, motor, battery holder, and all necessary parts to build your own wooden robot, complete with blinking eyes. Includes easy-to-follow instructions. Requires screwdriver and long nose pliers for construction, plus two AA batteries - all not included. Recommended for ages 10 and up.



3152147 \$19.95
Robomech Wooden Kit

WARNING:
CHOKING HAZARD—Small Parts.
Not for children under 3 yrs.

POW Robot T-Shirt

From R2-D2 and C3-PO to Johnny 5 and Rosie, there are few things cooler than robots. Now you can proudly sport your robot fandom in this cotton tee. Accented by comic book style POW background, the robot pictured on this shirt will WOW your friends while you look cool, calm, and comfortable. Made in America and printed with environmentally friendly inks. Color: Heather

NEW



POW Robot : \$24.95

3200150.....Medium
3200151.....Large
3200152.....X-Large
3200153.....XX-Large

Virtual Guard Dog Protects Your Home 24/7



Dog not included

Switch from barking to tranquil rainforest sounds for a soothing audio backdrop for a party!

1320672 \$89.95
Rex Plus, The Electronic Watchdog

- Adjustable volume & sensitivity
- 5 1/4" L x 5 3/4" W x 7 1/2" H; 6' cord

Few security systems are as intimidating as an angry barking dog. The Rex Plus security system "sees" through walls and doors to detect when someone approaches your home and then it "barks" just like a real German Shepherd. The startlingly realistic barking increases in intensity as the interloper gets closer. You get inexpensive, reliable, 24-hour protection that's perfect for extended absences.

Expecto Patronum!



NEW

3200103 \$89.95
The Wizard's Wand Universal Remote

Cast A Magic Spell Over Your Electronics!

To truly rule the room and couch kingdom you must have a wizard on your side...or better yet, be a wizard yourself. With this vibrating wand as universal remote, controlling your home entertainment systems is simple sorcery. A flick of the wrist and a spin of your magical wand changes the channel, volume, track, and more, including rewind and fast forward. A total of 13 programmable commands are controlled by circular movements, up and down gestures, or back and forth whisks of this vibrating wand. It looks like magic, but it's really technology — the same accelerometer technology used in Wii remotes. With practice you'll master a level of wizard like skill to impress all your friends. The remote is recognized by almost every piece of modern home entertainment apparatus. Size: 39cm x 6.5cm x 3.8cm. Weight: 295g.

SERVO

MAGAZINE

01.2011

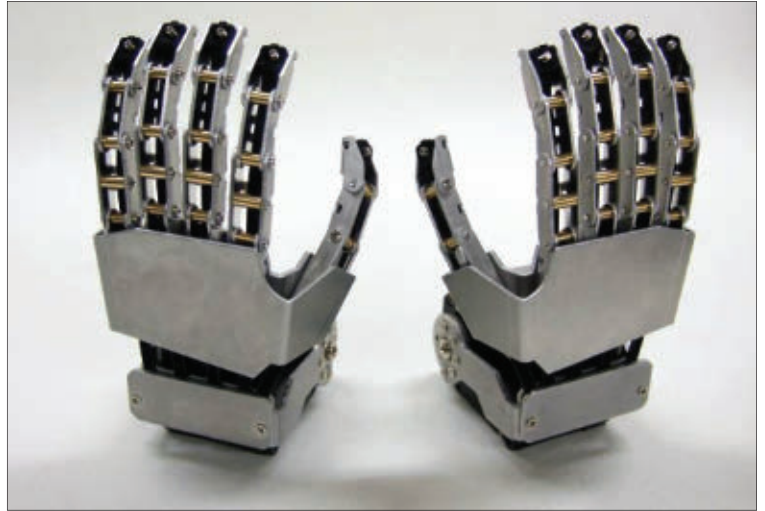
VOL. 9 NO. 1

Did you know that each article in *SERVO Magazine* has its own webpage? It's where you go for downloads, comments, updates, corrections, or to link to the article in the digital issue. The unique link for each webpage is included with the article.

You can also visit article pages from back issues at www.servomagazine.com. Just select the **Past Issues** tab from the **About SERVO** drop down menu, click the Table of Contents link, and the article name.

Columns

- 08 Robytes**
by Jeff Eckert
Stimulating Robot Tidbits
- 10 GeerHead**
by David Geer
Getting A Grip: Fingerless Robot Hand Grabs, Carries, Pours, and Writes
- 14 Ask Mr. Roboto**
by Dennis Clark
Your Problems Solved Here
- 70 Twin Tweaks**
by Bryce and Evan Woolley
Cirque Du Robot
- 76 Then and Now**
by Tom Carroll
Robot Hands


PAGE 76

The Combat Zone...

Features

- 22 PARTS IS PARTS:**
Identifying Unknown Brushed DC Motors
- 23 Affordable 2.4 GHz**
- 27 The Safe Use of Lithium Polymer or Lithium-Ion Batteries in a Combat Robot**
- 30 RioBotz Combot Tutorial Summarized: DC Motors**



- 35 Hello Bradley — Why Gambling Doesn't Pay**

Events

- 35 Results and Upcoming Events**

Departments

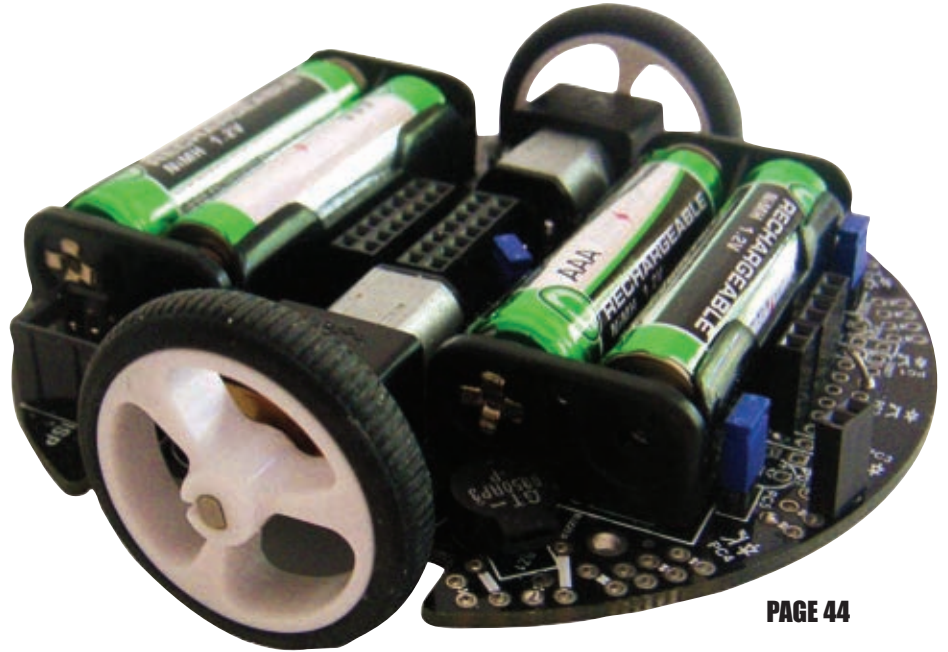
- 06 Mind/Iron**
- 07 Bio-Feedback**
- 18 Events Calendar**
- 19 Showcase**
- 20 New Products**
- 37 Bots in Brief**
- 67 SERVO Webstore**
- 81 Robo-Links**
- 81 Advertiser's Index**

SERVO Magazine (ISSN 1546-0592/CDN Pub Agree#40702530) is published monthly for \$24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

In This Issue ...



PAGE 40



PAGE 44

40 ComBots Cup V — Big Bad Bots Bash Bumpers in the Bay Area

by Dave Calkins and Simone Davalos

See the big-time participants and the ultimate winner in this classic clash of the titans.

44 A New Paradigm in Hobby Robotics

by John Blankenship and Samuel Mishal

Many industries successfully use the “simulate then deploy” principle. Apply it to your next robot build and you’ll soon see the advantages to this technique.

49 Experience the Emperor Pinguino

by Fred Eady

Pinguino is an Arduino-like board based on a PIC microcontroller. This article explores some of the features of the EDTP Electronics Emperor version ... and only hits the tip of the iceberg.

54 The NXT Big Thing #6

by Greg Intermaggio

Meet Eddie 2.0 — the next generation of LEGO MINDSTORMS design.

60 Making Robots With the Arduino — Part 3

by Gordon McComb

Inside the Arduino. This time, we’ll learn more about the Arduino and its programming.



PAGE 54

Mind / Iron

by Bryan Bergeron, Editor



Government Push for Robotics

Advances in robotics, like the growth of the economy, seem to be at a standstill. As a result, few new technologies have trickled down to the enthusiast level. That's about to change. The Defense Advanced Research Projects Agency (DARPA), the National Institutes of Health (NIH), National Science Foundation (NSF), United States Department of Agriculture (USDA), and the Department of Homeland Security (DHS) have teamed up to fund a 100 small business grants that will advance the field of robotics. You can find the full announcement at <http://grants.nih.gov/grants/guide/pa-files/PAR-10-279.html>.

This is great news for the field of robotics. Because so many agencies are involved, the areas of robotics that will be funded are exceptionally broad. In fact, the most intriguing part of the grant announcement is the range of categories eligible for funding. Reproducing the full list of categories here would require several pages. Even so, the main categories and example topic areas are worth reviewing because they provide an encyclopedia view of the field of robotics. I've arranged notable examples of robotics research topics by funding agency.

National Institutes of Health (NIH), the primary US Federal agency for conducting and supporting biomedical and behavioral research, is funding research directed towards innovations in robotics in four main areas: home care; rehabilitation; surgery; and laboratory automation. In the area of home care is funding to develop robots that remind patients to take their medication, monitor blood pressure and other vital signs, and wirelessly communicate the findings to a doctor or nurse.

Rehabilitation robotics range from robot caregivers, robotic implants and prosthetics, and devices that help the blind navigate, to avatar-based psychiatrists. The flashiest topics in the area of robotic surgery include implantable smart robots, robots to train surgeons, robotic surgical assistants, and organ/limb replacement robots. Some of the most promising areas of development in laboratory automation robotics include the remote sensing of toxins in the environment and the automated storage and retrieval of medical samples in storage. For humans, the first task is potentially harmful; the second is simply boring.

The **Defense Advanced Research Projects Agency (DARPA)** which is tasked with maintaining the technological superiority of the US military, is promoting the development of actuators that can replace human muscle. DARPA is looking for 'better than biology' actuators that meet or exceed the safety and efficacy of human muscle. I'm thinking Terminator 3 is a good place to start.

The **National Science Foundation (NSF)**, a long-time source of funding for the robotics community, is promoting the development of technologies that support patient mobility and rehabilitation robotics. The NSF has promised support for fundamental research in materials, manufacturing, signal processing, micro electromechanical system (MEMS) devices, neural control, social-assist robots, simulators, robots for training/learning processes, and energy harvesting techniques.

The **United States Department of Agriculture (USDA)** which is focused on food, agriculture, and natural resources, seems out of place in this group of funding agencies. However, the agency is looking for robots to assist in the production, harvesting, sorting, storing, processing, inspection, packaging, marketing, and transportation of food. These are no easy tasks from a robotics perspective. Consider what's involved in picking fruit from a tree or bush, from image recognition and fine motion control, to the ability to work in an outdoor environment that may be hot, cold, sunny, dark, wet, or dry. Or, think of what's involved in vaccinating a calf or other uncooperative animal.

The **Department of Homeland Security (DHS)**, saddled with preventing terrorism and enhancing security, is promoting robotic technologies to counter improvised explosive devices (IEDs) and for cross border tunnel surveillance. In brief, DHS is looking for technologies to remotely or robotically access, diagnose, and render safe IEDs, and to remotely survey tunnels and other subterranean structures. One of the interesting caveats is that the survey robots should be capable of "rapidly traversing obstacles such as stairs, curbs, and corrugated drainage pipes." I can't wait to see what someone comes up with to handle corrugated drain pipes.

FOR THE
ROBOT
INNOVATOR

SERVO
MAGAZINE

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct., Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore Only **1-800-783-4624**
www.servomagazine.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

**ASSOCIATE PUBLISHER/
VP OF SALES/MARKETING**
Robin Lemieux
display@servomagazine.com

EDITOR
Bryan Bergeron
techedit-servo@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert
Tom Carroll
Dennis Clark
Fred Eady
Bryce Woolley
Gregory Intermaggio
Gordon McComb
David Calkins
Pete Smith
Bradley Hanstack
Jenn Eckert
David Geer
R. Steven Rainwater
Kevin Berry
Evan Woolley
John Blankenship
Samuel Mishal
Simone Davalos
Steven Nelson

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

**MARKETING COORDINATOR
WEBSTORE**
Brian Kirkpatrick
sales@servomagazine.com

WEB CONTENT
Michael Kaudze
website@servomagazine.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION/GRAPHICS
Shannon Christensen

Copyright 2011 by
T & L Publications, Inc.
All Rights Reserved

All advertising is subject to publisher's approval. We are not responsible for mistakes, misprints, or typographical errors. *SERVO Magazine* assumes no responsibility for the availability or condition of advertised items or for the honesty of the advertiser. The publisher makes no claims for the legality of any item advertised in *SERVO*. This is the sole responsibility of the advertiser. Advertisers and their agencies agree to indemnify and protect the publisher from any and all claims, action, or expense arising from advertising placed in *SERVO*. Please send all editorial correspondence, UPS, overnight mail, and artwork to: **430 Princeland Court, Corona, CA 92879.**

Printed in the USA on SFI & FSC stock.



Funding begins in September of 2011, and a typical small business grant or SBIR provides support for up to a year. With all that activity in robotics, it's inevitable that we'll have a new generation of hardware, software, and robotics systems to work with. If you or your company is competing for one of these grants, please consider sharing a high-level description of your work with our readers. **SV**

Dear *SERVO*:

On Page 47 in the November '10 issue, the diodes are installed backwards or it's a typo.
Joe Santana

Good catch! The diode is backwards in both the schematic and the diagram. The diode is reversed biased so that when the magnetic field in the relay collapses, the back EMF current it creates shorts through the diode and does not flow back into the microcontroller operating the relay.
Paul Verhage

**MOTORS
GEARBOXES
WHEELS
AND MORE**

BB BaneBots **BANEBOTS.COM**
970-461-8880

BLFNARDWEENY! ...wait, what?



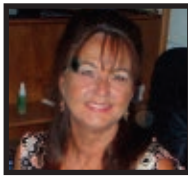
SKU: KARDWBP
Price: \$11.95

The **Ardweeny Backpack** adds full portability, 5V regulation, Servo, Temperature & Blink-M features to the Arduino-compatible Ardweeny! Now with fancy-schmancy **Blinky Light For No Apparent Reason** technology.

SOLARBOTICS
www.solarbotics.com 1-866-276-2687

PS- What else would it stand for? Bacon Lettuce Feta Avacado Raddish Dill Wheat...eeny?





Robytes

by Jeff and Jenn Eckert



Artist's concept of the Transformer vehicle. Source: DARPA.

Specs call for the vehicle to travel at least 250 nautical miles in any land/air combination while carrying up to 1,000 lb. Phase 1 involves 12 months of conducting "trade studies to develop and mature propulsion systems, adaptable wing structures, advanced lightweight materials, the advanced flight control system, the air/ground configuration designs, and energy distribution systems." If all goes well, the project will move on to an unspecified second phase.

Real Transformers in Phase 1

No, it's not an illustration from a toy package or a movie poster. This is the desired end product of the Transformer (TX) program kicked off by the folks at the Defense Advanced Research Projects Agency (DARPA, www.darpa.mil) last fall. Some \$3 million in funding has been awarded to several research participants — notably AAI Corp. (www.aaicorp.com) and Lockheed-Martin (www.lockheedmartin.com) — to look into the concept of building a sort-of Humvee/jump jeep/gyroplane concoction that can travel over any kind of terrain. Basically, a CarterCopter driven by a gas turbine propeller and fitted with foldable wings and rotor mast. The guidance and flight control systems are to allow semi-autonomous flight, so even someone with no pilot training can perform VTOLs, transition to forward flight, and modify the flight path as needed.

This Gripper Sucks



This robotic gripper conforms to the shape of virtually any object. Photo by John Amend.

Every once in a while an idea comes along that's so simple it's brilliant, and a case in point is a universal robotic gripper emerging from researchers at Cornell University, the University of Chicago, and iRobot Corp. Whereas most gripper designs tend to be variations on the human hand and fingers, this one is basically a vacuum cleaner attached to a balloon filled with ground coffee. In operation, you just press the balloon against an object which causes it to deform and fit around a portion of the target. You then

suck the air out of the balloon, and the coffee undergoes a "jamming transition" that turns its behavior from that of a fluid to that of a solid, i.e., the grounds can no longer slide past each other. The result is a gripping action that works with virtually any shape. Releasing the object is simply a matter of turning off the vacuum. According to Prof. Hod Lipson, the universality of the gripper makes future applications seemingly limitless, including implementation as feet on a bot that can walk up walls.

So, why didn't you think of that?

For a more detailed explanation of this gripper, check out the GeerHead column in this issue.

Free Robot Art



My Red Tie, one of many pieces of downloadable bot art.

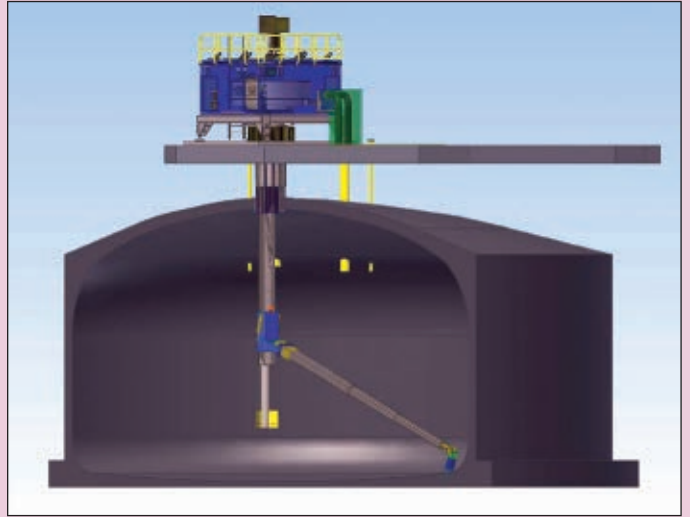
As occasionally noted in these pages, when artists and robots tangle, the bots usually come out on top. An exception, however, is some pretty neat images collected and posted by graphic designer Eric Shafer at creativefan.com/40-brilliant-robot-artworks/. No copyright information is included for most of them, but it's a fair guess that you can download anything you like and use it as a desktop theme or something like that.

Bot Handles Nuke Waste

Way back in 1943, the Department of Energy set up nine nuclear reactors at the Hanford Site (www.hanford.gov) in the desert of southeastern Washington. The reactors were used to generate plutonium for atomic weapons from WWII through the Cold War. Although the reactors were shut down in 1987, the 586 square mile facility is still the home of billions of gallons of liquid waste, and millions of tons of solid radioactive materials. The stuff is stored in underground tanks, 16 of which — unfortunately — are old single-shell tanks that are prone to leaking and must be drained before they cause serious trouble.

This isn't the kind of thing you want to do with a sump pump and a garden hose, and the process is made more difficult by the relatively narrow 12 inch diameter pipe that's available to access the contents. However, a solution apparently has been devised by Washington River Protection Solutions (www.wrpstoc.com), a DoE contractor. The fix involves replacing the 12 inch riser with a 42 inch one, allowing a robotic arm — known as the Mobile Arm Retrieval System, or MARS — to be installed inside. They are beginning with a tank known as C-107, where the bot will use a water cannon to move waste towards a pump which will suck the gunk out at rates ranging from 85 to 1,000 gallons per hour, depending on the type of waste.

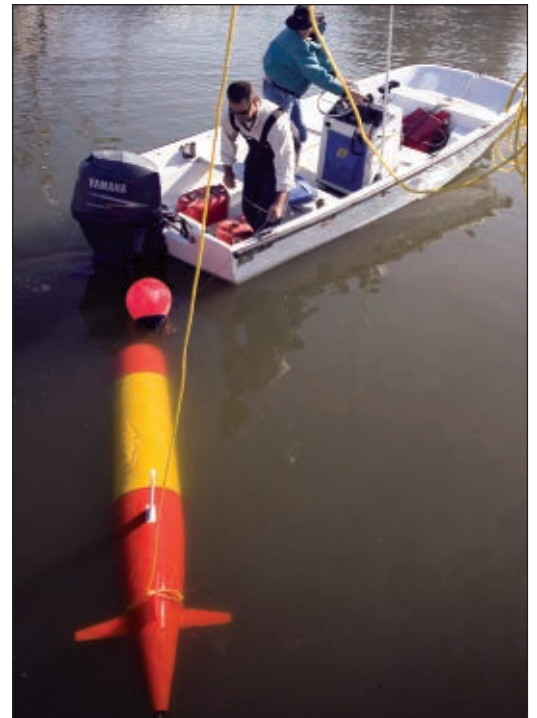
According to WRPS, this will allow the 247,000 gallons to be removed quickly enough to avoid exposing workers to any more than about 40 millirems per hour, keeping them within the administrative limit of 500 millirems per year. A tougher nut to crack will be C-111 which contains a stubborn crust of waste that has fended off attacks with a high pressure sprayer. But, one step at a time ...



Artist's rendition of the Hanford tankbot.

AUV Combines Propulsion Schemes

Until recently, there have been essentially two types of autonomous underwater vehicles (AUVs): relatively fast-moving, propeller-driven units that can carry a good-sized load of instruments but are limited to stints of only a few days' operation; and "gliders" which can operate for months at a time but are slow and carry a limited payload. But a new seabot called Tethys combines the best of both these worlds. In high-speed mode, it can travel at about 2.25 mph (1 m/sec) which is about quadruple the speed of most gliders. It can also drop into hover mode, allowing it to remain in operation for a matter of weeks. Tethys is the work of the Monterey Bay Aquarium Research Institute (www.mbari.org) which has been working on it for about four years. Back in October, the first one was successfully tested in Monterey Bay, equipped to study phytoplankton blooms. It is hoped that the design will prove to be inexpensive enough for wide adoption in a variety of seagoing research and monitoring. The original Tethys, by the way, was a Greek aquatic goddess, daughter of Uranus and Gaia. She was both sister and wife of Oceanus, but we won't worry about that for now. **SV**



MBARI researchers tow Tethys behind a small boat in Moss Landing Harbor. Todd Walsh© 2010 MBARI.



GEARHEAD

by David Geer

Contact the author at geercom@windstream.net

Getting A Grip Fingerless Robot Hand Grabs, Carries, Pours, and Writes

What can you do with a latex balloon, some coffee grounds, and suction? You can create a universal jamming robotic gripper hand that can pick up anything. This unconventional approach to an end effector/manipulator skirts the many issues that come with developing a humanoid robot hand.

With multi-fingered humanoid hand research, there are challenges such as how to actuate the many finger joints, how to apply force sensing so delicate objects can be held firmly without breaking or damaging them, and how to address the great mass of algorithms and computations necessary to calculate the force applied to each object by each finger.

As such, fingered hands are extremely intricate due to hardware complexities, actuation complexities, and the need for expansive software-based intelligence to perform the sub-tasks that lead to gripping and lifting with fingers.

As a result, humanoid finger-based hand research and development has proven expensive. Roboticians have invested themselves heavily to create universal gripping hands — hands that can pick up anything — using this model.

On the other hand, researchers and roboticians at Cornell University, the University of Chicago, and iRobot Corporation, have applied mechanical engineering and particle science to the complex problem of a universal robot gripper. Between them, Cornell's John Amend and Hod Lipson, Chicago's Heinrich Jaeger, and iRobot's Chris Jones

See how the universal jamming gripper safely grasps and lifts an egg without breaking it. (Now, if they can only teach one or two hands to crack the egg and whip up a nice dish ...)

Here, the hand grips and lifts a spring device (such as an actuator or shock absorber) which has a less predictable surface to work with.



have come up with a hand that is not human-like at all.

With their granular gripper, the mass of latex and coffee grounds surrounds and presses an object to get its grip, then a vacuum is created to harden the coffee grounds and latex in the new shape, securing the gripper's control of the item.

Turn Up the Jams

Researchers based the universal jamming gripper on a jamming phase: "If a collection of granular material is loosely packed, it can flow like a liquid. If, instead, the granular material is more densely packed, it behaves like a solid, and we call this the jammed state," explains Eric Brown, postdoctoral scholar from the University of Chicago.

"The transition between these two states is very sharp and is called the jamming transition. The difference in density between the two states is very small. This is analogous to the sharp phase transition between water and ice if the temperature is reduced," Brown illustrates.

With the surface of the gripper covering only one-fourth of the surface (sometimes less) of an object, the attending vacuum can tighten the hand on or around a cup, pen, screwdriver, or most anything. The hand comes up against the object, taking shape around it. Once the hand is in place, the vacuum or suction is applied and the grasp of the gripper is complete.

"For our tests, everything was pre-positioned because we were not testing any sensors. We did some tests on the tolerance to positioning and found the gripper had no loss of gripping capabilities if it was off target by as much as a quarter of the gripper size," says Brown. "For autonomous robot applications, this suggests even something rudimentary like a web cam may be all the sensing that is necessary. For other applications like assembly line or manually operated grippers, no sensing may be necessary at all," Brown notes.

Adjust the Volume Slightly

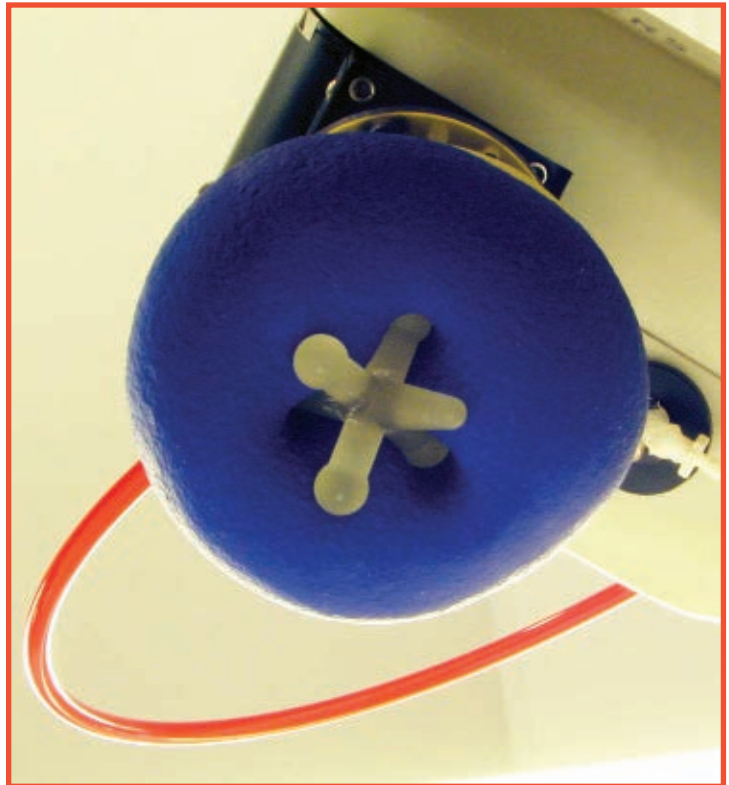
By decreasing the overall volume of the balloon by as little as one half of one percent, the roboticists can turn the malleable balloon into a rigid form around part of an object that is enough to grip and lift it up. "There is less space between the particles of coffee grounds and less space volume inside the balloon," says Brown.

Just a small amount of vacuum is enough to turn the soft, pliable latex and coffee grounds into a rigid confine about the item's surface. The gripper can hold and carry multiple items that are in close proximity to one another, as well. A pump known as a Venturi aspirator sucks the air out of the gripper. A red tube runs from the gripper to the pump to accomplish this.

"We used a variety of pumps for different



The unique latex membrane and coffee particle robotic hand is able to grip this glass and pour liquid from it.



The gripper hand is ready to play a round of jacks, but uses only one jack at a time for now.

GEERHEAD



Here, the insides of the advanced, balloon-like gripper are only coffee grounds. Amazing how much roboticists can do with simple components.

prototypes. We focused on the gripper because different pumps could be used for different applications," says Brown. The connection to the pump requires a filter to prevent the coffee grounds from leaving the gripper.

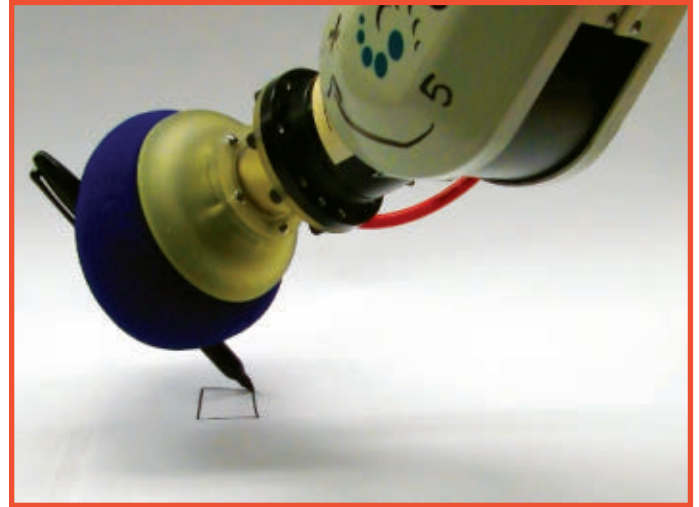
The scientists attached the gripper to a robotic arm in a fixed position. The arm selected for the gripper is a CRS A465 model. "The model is convenient for pick and place testing. We only performed tests to evaluate the grip performance, and not to evaluate characteristics with a specific arm, as different arms could be chosen for different applications," says Brown.

The gripper can lift items many times its own weight and size. This is due in part to the geometry involved when the gripper presses so firmly against the object's surface. It is also because of the friction between the gripper and the object. "The gripper picks up objects using friction when it pinches them, even if it doesn't wrap very far around those objects. This is similar to someone palming a basketball with one hand," explains Brown.

Apply Here

This balloon and coffee ground end effector will be ideal in applications where the robot arm and hand have to pick up multiple items together, especially items which the robot has not previously seen. Specific scenarios include military robots for removing bombs, robot maids or butlers for picking up items in the home, and picking parts in industrial settings where items on a conveyor belt are not precisely positioned.

"Generally, we think our gripper concept would work well where a variety of objects need to be gripped, but where sensing and computation are difficult or expensive, or ease of use is desirable," says Brown. Examples of such applications include hazardous materials work, remote controlled robots for search and rescue missions, prosthetics, and working under water.



Your autograph? As intelligent as it is practical, the hand holds a pen and writes on a surface.

The researchers intended this first work on the gripper to establish a foundation for a jamming gripper and to create a model of gripping characteristics for such an end effector. The model will help roboticists design future grippers enhanced for efficiency for specific applications under specific conditions. "Now, we are looking for partners who want to productize the technology," says Brown.

Conclusion

With all the complexities involved with a humanoid hand, it's amazing how a simple yet out of the box answer may be the right one after all. **SV**

Photos by John Amend, Cornell University.

Resources

Gripper home page
http://ccsl.mae.cornell.edu/jamming_gripper

Universal gripper video
www.news.cornell.edu/stories/Oct10/Hod.Lipson.gripper.mov

Eric Brown home page
<http://home.uchicago.edu/~embrown>

John Amend home page
www.johnamend.com

Hod Lipson home page
www.mae.cornell.edu/lipson

Heinrich Jaeger home page
<http://jfi.uchicago.edu/~jaeger/hmj>

iRobot Corporation
www.irobot.com

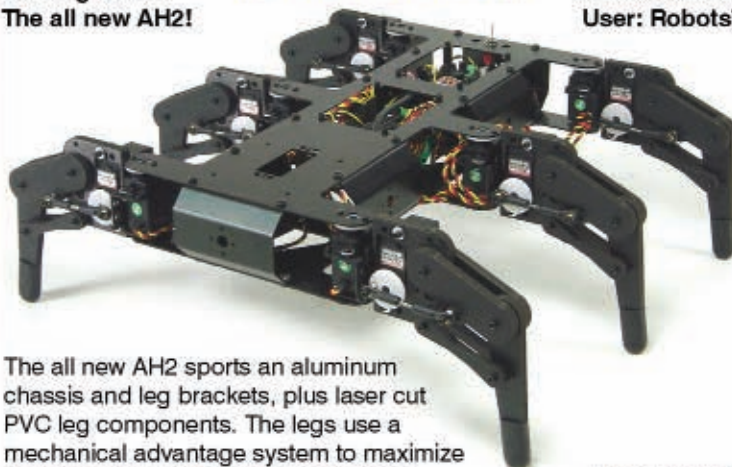


**The Lynxmotion Servo Erector Set
Imagine it... Build it... Control it!**

Featured Robot

**Coming Soon!
The all new AH2!**

**Youtube videos
User: Robots7**



The all new AH2 sports an aluminum chassis and leg brackets, plus laser cut PVC leg components. The legs use a mechanical advantage system to maximize payload even when using inexpensive servos. This is 2DOF Hexapod, done right!

**Black or clear
anodized finish!**



Biped Nick



Biped Pete



Biped Scout



Biped 209



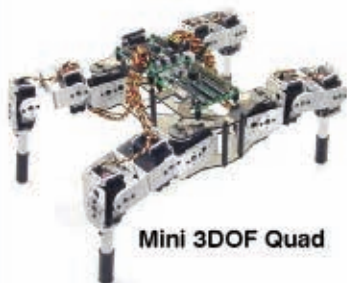
Walking Stick



CH3-R



T-Hex



Mini 3DOF Quad

**With our popular Servo Erector Set you can easily
build and control the robot of your dreams!**

Our interchangeable aluminum brackets, hubs,
and tubing make the ultimate in precision
mechanical assemblies possible.



All New ARC-32 - \$99.95
32 Channel Servo/Microcontroller.
USB Programming port.
32 bit Hardware based math.
Program in BASIC, C, or ASM
Servo and Logic power inputs.
Sony PS2 game controller port.



Bot Board II - \$24.95
Carrier for Atom / Pro, BS2, etc.
Servo and Logic power inputs,
5vdc 250mA LDO Regulator.
Buffered Speaker.
Sony PS2 game controller port.
3 Push button / LED interface.



SSC-32 - \$39.95
32 Channel Servo Controller.
Speed, Timed, or Group moves.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
TTL or RS-232 Serial Comms.
No better SSC value anywhere!

We also carry motors, wheels, hubs, batteries, chargers,
servos, sensors, RC radios, pillow blocks, hardware, etc!



Visit our huge website to see our complete line of
robots, electronics, and mechanical components.



Biped BRATs



Phoenix





Our resident expert on all things robotic is merely an email away.
roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

ASK MR. ROBOTO

by
Dennis Clark

Welcome to 2011! It's a new year with even more amazing advances in robotics which — for some reason — no one but us roboteers ever seem to know about! No matter. I guess that means there are just that many more people out there to amaze with our hobby! As I write this, there is a lot of buzz about the Microsoft Xbox 360 add-on "Kinect" sensor which was hacked within a week of its release. It can be used to do 3D mapping of its environment, as well as all the cool stuff that an Xbox 360 does with it. Neat! Another sophisticated sensor to add to the toolbox! Keep watching those hacker blogs!

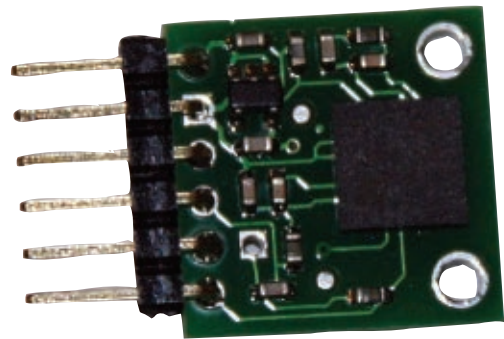
Q I want to know that when my robot is falling down, all I need is an accelerometer, right? My robot is just a basic walker that uses servos to move. Any help you can offer would be great!

— Kevin

A Kevin, well, not really. It is true that an accelerometer will allow you to detect your robot tilting over. Unfortunately, as the name "accelerometer" implies, the device measures acceleration. It can't distinguish between a change in attitude measured by the change in the effects of gravity, in your robot, or the moment-to-moment accelerations caused by your robot simply moving. You will need one more device to help you with your "tilt" sensor: a gyroscope, or "gyro" for short.

The accelerometer can measure a change in the effects of gravity which can indicate a tilt, but to make sure that you are tilting and the measurement isn't just your robot shambling, you can use the output of your gyro to confirm the deduction. A gyro will measure a change in rotation around a given axis. So, if you orient your gyro so that its axis is perpendicular to the axis of the tilt you are interested in (this tends to be the front-to-back vector), then when your accelerometer sees a change that the gyro confirms you can measure the two of them to determine how much change there is and attempt an adequate amount of

Figure 1. Gyro breakout board.



correction to stabilize your walker.

Since I've been in a "code writing" kind of mood these last few months, I've put together an example piece of code that will read the angle reported by an RC aircraft hobby gyro. I know that you can get gyros that read out in just plain analog signals like the Pololu **LISY300AL** (Figure 1) these days, but I was convinced that the RC gyros have better stability and drift less, so I wanted to try them out. Unlike the Pololu gyro though, the output of the GWS **PG-03** wasn't an analog voltage, but a servo pulse modified by the



Figure 2. GWS PG-03 RC gyro.

magnitude of the angle change. This means that I need to read a pulse width. (Goody! Something new!) We've seen how to generate a servo pulse using an interrupt in previous *Roboto* columns, now let's learn how to read a pulse width on an Atmel ATmega 168 chip.

The PG-03 gyro costs about \$38.50 at **Tower Hobbies** and, for instance, a Pololu gyro breakout board like the one shown in **Figure 1** is only \$19.95. However, the PG-03 is temperature stabilized and has a gain adjust so you can increase or decrease its response to rotational changes (see **Figure 2**). The PG-03 has a traditional hobby servo connection as its input and another traditional hobby servo connector for an output. If we want to measure our rotational change, we need to measure the pulse width coming out of the gyro.

I'm going to use my typical ATmega 168 board with my usual boilerplate setup code to demonstrate how to measure the PG-03 output. The fastest way to measure a pulse is to use hardware that is built into a microcontroller rather than trying to measure something in a code loop. A code loop can measure what we want, but everything else that we are doing in our robot will stop while we are in that code loop; we want to be able to measure these changes fast, so we use an interrupt based *Input Compare* module attached to *Timer 1*.

Listing 1 shows the code I used to set up the interrupt and **Listing 2** is the *Interrupt Service Routine* (ISR) code that does the actual measurement.

The register **TCCR1B** has the bit settings required to configure the

Input Compare functionality of Timer 1. I have set bit 7, *ICNC1* (Input Noise Canceller), which delays the signal four clocks, but makes sure that the input is stable. Also set is the *ICES1* bit which tells the Input Capture module to interrupt on the rising edge of the pulse. Finally, I chose the full speed clock (bits 2:0) so that the full scale reading in 16 bits equals 4.096 ms. This gives us a very good

Listing 1: Input Compare setup.

```
void InitTimer1(void)
/*
 * Set up the ICP1 input compare interrupt system.
 *
 */
{
    TCCR1A = 0x00;           // No Output compares
    TCCR1B = 0xC1;           // rising edge, noise canceler
    TCCR1C = 0x00;           // just to be thorough
    TCNT1H = 0x00;           // clear the counter
    TCNT1L = 0x00;

    edge = 1;                // flag rising edge half
    TIMSK1 |= (unsigned char)_BV(ICIE1);    // enable interrupt
}
```

Listing 2: Input Capture ISR.

```
ISR(TIMER1_CAPT_vect )
{
    if (edge == 1) {
        TCNT1H = 0;
        TCNT1L = 0;           // we want the positive side pulse
width
        edge = 0; };           // Now look for falling edge
        TCCR1B &= ~(unsigned char)_BV(ICES1)
    }
    else {
        pulse = (uint8_t)ICR1L;
        pulse += (uint16_t)(ICR1H <<8);    // get width
        edge = 1;
        TCCR1B |= (unsigned char)_BV(ICES1); // flip trigger sense
    }
}
```

Listing 3: Volatile definition for ISR variables.

```
/* Global stuff to use */
volatile uint8_t    servo;           // My one and only servo
volatile uint32_t   t_1ms;           // Background 1ms ticker
volatile uint8_t    edge;            // CIP1 capture edge
volatile uint16_t   pulse;           // our pulse width
```


Listing 4: Main code loop.

```
while(1)
{
    waitms(100);                // once every 100ms
    if ((abs(pulse - lastPulse)) > 100)
    {
        lastPulse = pulse;
        printf("%u\r",pulse); // print out the pulse value
    }
}
```

measurement resolution.

In avr-gcc, there is a macro called *ISR()* which takes the interrupt vector as an input; the documentation for avr-gcc tells the vector names, but you can just as easily find the interrupt vector name by looking at and using the Atmel ATmega 168 datasheet in section **11.4: Interrupt Vectors in ATmega 168** which lists vector 11 "TIMER1 CAPT." Modify this name like this: `TIMER1_CAPT_vect` and you have the vector to pass to the *ISR()* macro. Our ISR needs to handle both the interrupt for the rising edge of the pulse and the interrupt for the falling edge of the servo pulse. You will notice that on the rising edge I clear the 16-bit *TCNT* register; this allows our timing to start at zero. I then say that the next interrupt will occur on the falling edge by clearing the *ICES1* bit. When the servo signal drops, the *else* clause of the ISR is used. This gets the value of the *ICR1* register which is the value of the *TCNT* register at the

time of the interrupt. I then flip the *ICES1* bit to again interrupt on a rising edge and the cycle starts over again.

When you use an ISR to set or change a variable in your program, you need to let the compiler know this. This is because an ISR is never called from within your program so the compiler does not know when the variables that the ISR changes actually change. We do this by using the *volatile*

qualifier to the variable definition as shown in **Listing 3**. Because of the nature of these "volatile" variables, they are global in scope. I can see you "big iron" programmers cringing at the "global" definition, but this is embedded programming. Many of the rules change "down here!" In "programmer speak," *volatile* tells the compiler to always read the memory location for that variable; never assume it hasn't changed.

Now let's look at the *main()* code where we actually use the value of the gyro that we are measuring. In **Listing 4**, we see the very short code section that determines the pulse value has changed and prints it out.

Notice that the code just assumes that the value is there in the **pulse** variable. Our ISR handles the dirty work and our code just uses the value. This loop tracks the previous pulse value and only prints something when it changes significantly. This keeps our display from filling up and blasting by us, and makes the changes easier to detect. Speaking of which, **Figure 3** shows my terminal window. I am a hard-core Mac user, so if you are a Windows user,

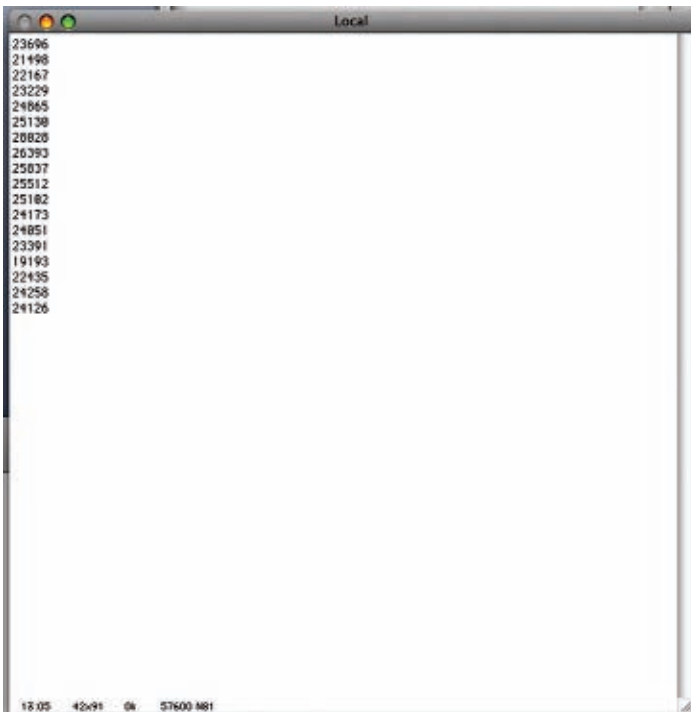


Figure 3. Gyro readings.

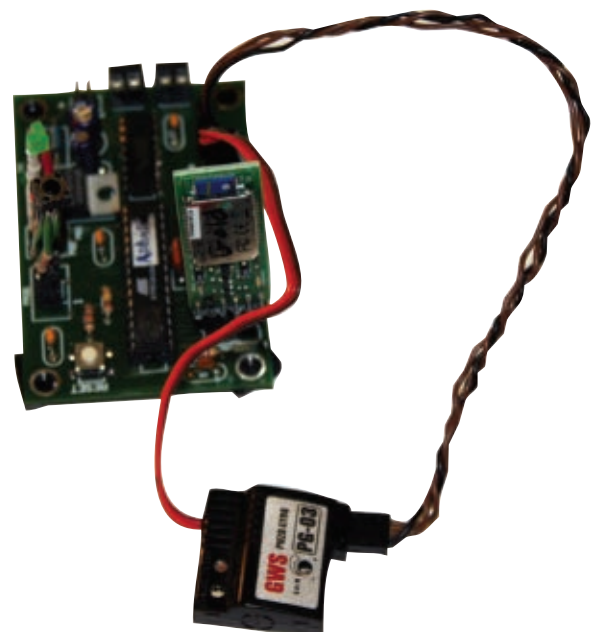


Figure 4. Test setup.

that is why you don't recognize the terminal type. You'll have to tweak your code to put out the correct line-terminating character to get a single reading on a single line with your favorite terminal emulator.

Notice the readings. They range from 19193 to 28028. These are timer clocks in an Arduino's 16 MHz system. With each clock being 62.5 ns, this is a servo pulse range of 1.2 ms to 1.75 ms. I was rotating the gyro back and forth to get these readings. **Figure 4** shows my setup. I generate a servo pulse of 1.5 ms on PD7 which is fed to the PG-03 gyro and reads the results back on ICP1 (PB0).

If you are using a gyro similar to that shown in **Figure 1**, you could still use an interrupt to handle the readings. You would just use the vector 22 `ADC_vect` interrupt which would get the analog value of the output of the gyro breakout board. The same holds true for reading your accelerometer, if it's like the one shown in **Figure 5**. That's the Pololu MMA7260QT breakout board (recently obsolete and replaced by a newer device). Technology keeps changing and it constantly amazes me that between the time I buy a new widget and the time I use it, it's obsolete!

The source for the gyro program can be found at the *SERVO Magazine* website: under *Mr. Roboto* as *rcgyro.zip*. As always, if you have a question for Mr. Roboto, drop me a line at roboto@servomagazine.com and I'll be happy to try to answer it! Have fun and keep building robots! **SV**



Figure 5. Three-axis accelerometer.

Robot Kits



- Line followers
- Mini-sumos
- Robot arms

Save 10% with coupon code SERVO3PI26

\$99.95
Item #975

3pi Robot

High-performance, C-programmable, ATmega328P-based robot (with Arduino support)

Electronic Parts

- Sensors & controllers
- Discrete components
- Cables & batteries



Jrk USB Motor Controller with Feedback
\$49.95



QTR reflectance sensors
\$2.49



Micro Maestro 6-Channel USB Servo Controller
\$19.95

Custom Laser Cutting & Solder Paste Stencils

Mechanical Components

- Motors & servos
- Wheels & ball casters
- Chassis & more!



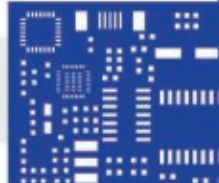
\$39.95
Metal gearmotor with 64 CPR encoder



Pololu Wheels
\$7.95 - \$9.95

Both starting at only \$25!

Use our low-cost plastic solder paste stencils to quickly assemble your surface-mount designs.



Laser cut your own custom chassis, front panels & more!



1-877-7-POLOLU
www.pololu.com

EVENTS

Calendar

ROBOTS.NET

Send updates, new listings, corrections, complaints, and suggestions to: steve@ncc.com or FAX 972-404-0269

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: <http://robots.net/rcfaq.html>
— R. Steven Rainwater

JANUARY

- 11** **First LEGO League of Central Europe**
*Heinz Nixdorf Museum Forum
Paderborn, Germany*
Regional European FLL teams compete for the National Championship.
www.hands-on-technology.de/en/firstlegoleague
- 25** **Powered by Sun**
Ostrava, Czech Republic
Solar-powered autonomous robot race.
<http://napajenisluncem.vsb.cz>
- 25-27** **Singapore Robotic Games**
Singapore Science Center, Republic of Singapore
Autonomous robots compete in 17 events that include Sumo, legged robot marathon, legged robot obstacle race, Micromouse, underwater, and more.
<http://guppy.mpe.nus.edu.sg/srg>
- 28-31** **Robotix**
IIT Khargpur, West Bengal, India
Robots of all kinds compete in events including Xants, TribotX, Xtension, 8mileX, ASME, Xplode, and eXplore.
www.robotix.in

FEBRUARY

- 2-5** **Kurukshetra**
Guindy, Chennai, India

Robotics events this year include K!onstructor, Khimera, Pandemonium, XCEED, and Designer Quest.

www.kurukshetra.org.in

- 17-20** **Pragyan**
NIT, Trichy, India
Events include Micromouse, Fix-the-Android, PIP bots, and Crop Circles.
www.pragyan.org
- 17-20** **Techkriti RoboGames**
IIT, Kanpur, Uttar Pradesh, India
This year's events include Reconnaissance, Robot's Got Talent, Trip to the Future, and FIFA 2050.
www.techkriti.org/#/competitions/robogames

MARCH

- 6-10** **APEC 2011**
*Fort Worth Convention Center
Ft. Worth, TX*
In this event, autonomous Micromouse maze running for bots 25 cm x 25 cm. Cash prizes and trophies.
www.apec-conf.org
- 11-12** **AMD Jerry Sanders Creative Design Contest**
University of Illinois at Urbana-Champaign, IL
This event includes autonomous and remote-control.
<http://dc.cen.uiuc.edu>
- 12-13** **RobotChallenge**
Vienna, Austria
This event includes Parallel Slalom, Slalom Enhanced, Standard Sumo, Mini Sumo, and Micro Sumo.
www.robotchallenge.org
- 22-24** **DTU RoboCup**
*Technical University of Denmark
Copenhagen, Denmark*
This event has a varied course including line and wall following.
www.robocup.dtu.dk

APRIL

9-10 Trinity College Fire Fighting Home Robot Contest

Trinity College, Hartford, CT

The goal of this contest is to produce a computer-controlled robot capable of navigating through a mock house, locating a fire, and extinguishing the flames. The robot with the shortest time wins.

www.trincoll.edu/events/robot

10 Robotics Innovations Competition and Conference

Woburn, MA

This event includes robot mobility through unconventional means.

<http://ricc.wpi.edu>

11-12 IEEE TEPRA Student Robotics Competition

Woburn, MA

See website for information.

www.tepra2011.wpi.edu

14-16 National Robotics Challenge

Marion, OH

See website for information.

www.nationalroboticschallenge.org

14-16 VEX Robotics World Championship

Kissimmee, FL

This event includes high school and university VEX contests.

www.vexrobotics.com/competition

15-17 RoboGames

Ft. Mason's Festival Pavillion, San Francisco, CA

This event includes FIRA, BEAM, MINDSTORMS and Combat.

www.robogames.net

23 Baltic Robot Sumo

Klaipeda, Lithuania

This event is mini Sumo. See website for more information.

www.balticrobotsumo.org

30 The Tech Museum of Innovation's Annual Tech Challenge

Parkside Hall, San Jose, CA

See website for information.

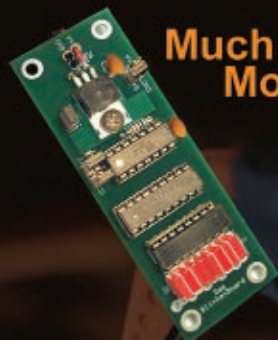
<http://techchallenge.thetech.org>

Robotics Showcase

Das Blinkenboard

Not just for
blinker LEDs.

Much Much
More!



Complete kits
available @

<http://store.nutsvolts.com>
&

<http://store.servomagazine.com>

**ALL
ELECTRONICS**
CORPORATION

THOUSANDS OF ELECTRONIC
PARTS AND SUPPLIES

VISIT OUR ONLINE STORE AT
www.allelectronics.com

WALL TRANSFORMERS, ALARMS,
FUSES, CABLE TIES, RELAYS, OPTO
ELECTRONICS, KNOBS, VIDEO
ACCESSORIES, SIRENS, SOLDER
ACCESSORIES, MOTORS, DIODES,
HEAT SINKS, CAPACITORS, CHOKES,
TOOLS, FASTENERS, TERMINAL
STRIPS, CRIMP CONNECTORS,
L.E.D.S., DISPLAYS, FANS, BREAD-
BOARDS, RESISTORS, SOLAR CELLS,
BUZZERS, BATTERIES, MAGNETS,
CAMERAS, DC-DC CONVERTERS,
HEADPHONES, LAMPS, PANEL
METERS, SWITCHES, SPEAKERS,
PELTIER DEVICES, and much more....

ORDER TOLL FREE
1-800-826-5432

Ask for our FREE 96 page catalog

Esduino12

- 9S12C 16-bit microcontroller in Arduino form-factor!
- 32K or 128K Flash
- optional USB interface
- low-cost Xbee plug-in option

Use with any
Arduino shield!



From
\$39

Easy object-based
Programming with nqBASIC!
Advanced programming in C
with CodeWarrior



Four new proto shields!

TechnologicalArts.com

NEW PRODUCTS

TELEMETRY

USB Telemetry Receiver

Hitec announces a new telemetry item: the HTS-Navi which takes their telemetry system to the next level of convenience and functionality.

This USB telemetry receiver lets you wirelessly download and display essential telemetry information on your PC without cables. Compatible with any radio fitted with their AFHSS system, the HTS-Navi will display everything you need to know about your model including: receiver or power battery voltage; fuel level; temperature; RPM; GPS; current; and voltage.



Telemetry Voice System

Hitec also announces their new telemetry voice announcing system: the HTS-Voice, which allows you to keep your eye on the sky and listen to what your model has to say.

This audible telemetry readout system is designed to fit on the handle of the Aurora 9, Optic 6, and Optic 6 Sport 2.4 GHz radios. When you fly with Hitec's telemetric Optima 7 or Optima 9 receivers, the system announces your model's data via a built-in speaker or optional earphones. Compatible with any radio fitted with their AFHSS system, the HTS-Voice operates on two AAA batteries and includes a volume adjust, convenient selectable information readout, a 3.5 mm earphone port, and a power LED indicator.

For further information on these items, please contact:



CONTROL SYSTEMS

Cross-link Robot Control System



Cross the Road Electronics, LLC introduces the Cross-link robot control system — the world's first and only CAN-based robotic control system. Cross-link consists of three primary components: the 2CAN Ethernet to CAN gateway; the CANipede RCM; and the TRENDnet wireless N pocket router.

2CAN — Two port Ethernet switch with a single channel two-wire CAN. The 2CAN is the backbone of the Cross-link system. It provides a gateway between Ethernet and CAN that allows viewing of critical system information such as current, voltage, sensor values, position, and PID values in an integrated web dash that may be accessed from a web browser. Boot time is in mS because the 2CAN firmware is embedded and does not suffer from the overhead of an operating system.

CANipede RCM (Robot Control Module) — The CANipede RCM is an open source hardware and software device that communicates with the 2CAN via CAN. The RCM provides a means of controlling PWM type speed controllers, direct control of solenoids, and H-bridge type relays. It also decodes quadrature type encoders directly, and contains analog inputs and general-purpose input/output (digital).

TRENDnet Wireless-n Router — This wireless router is as small as they come and supports three separate modes of operation. The 2CAN interfaces to the router via Ethernet to provide wireless control of your robot. The router has plenty of bandwidth to control your robot, view the 2CAN web dash, and view live web cam feeds.

The system comes pre-packaged at a discounted price of \$399.99. The components may be purchased individually, as well. All hardware, software, and firmware

are available from the website. Software and firmware are free downloads.

For further information, please contact:

**Cross the
Road Electronics**

Website:
www.crosstheroadelectronics.com

ROBOT KITS

New Drum Beetle Chassis Kit

Kitbots is now offering a new beetleweight fighting robot kit. Based on their successful, Weta, God of Ugly Things, it has pattern routed and jig drilled UHMW side walls and armor combined with watercut 7075 aluminum panels. Also available is a beater bar assembly for use with the chassis.



For further information, please contact:

KITBOTS

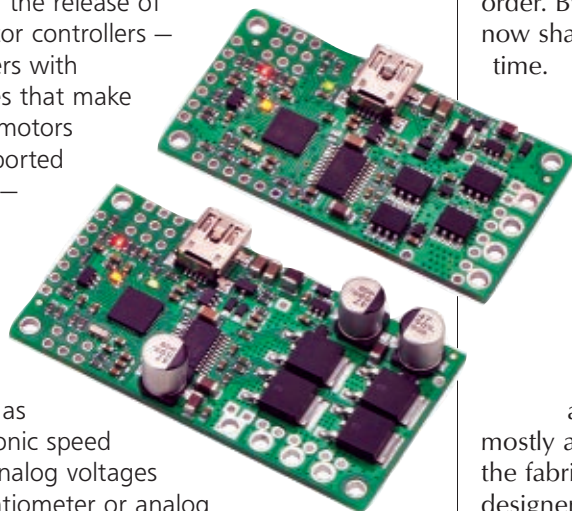
Website: www.kitbots.com

MOTOR CONTROLLERS

Simple Motor Controllers

Pololu announces the release of their simple motor controllers — a line of motor drivers with enhanced capabilities that make basic control of DC motors easy. With four supported high-level interfaces — USB for direct connection to a computer, TTL serial for use with embedded systems, RC hobby servo pulses for use as RC controlled electronic speed control (ESC), and analog voltages for use with a potentiometer or analog joystick — and a wide array of configurable settings, these devices simplify controlling motors in a variety of projects. Units can be paired to enable mixed RC or analog control of differential-drive robots, and they can be daisy-chained with additional Pololu servo and motor controllers on a single serial line.

A free configuration program (available for Windows



and Linux) allows for quick controller configuration over USB (no more DIP switches or jumpers) and simplifies initial testing.

Controller features include: acceleration and deceleration limits to decrease mechanical stress on the system; optional safety controls to avoid unexpectedly powering the motor; a wizard for automatic RC and analog input calibration; and support for limit switches.

The controller versions offer a wide operating voltage range up to 5.5-40V and continuous current ratings from 12A up to 25A, which means they can deliver up to several hundred watts in a small form factor. Unit prices are \$39.95, \$43.95, \$54.95, and \$59.95 for the 18v15 (item #1377), 24v12 (item #1379), 18v25 (item #1381), and 24v23 (item #1383), respectively.

For further information, please contact:

**Pololu
Corporation**

3095 E. Patrick Ln. #12
Las Vegas, NV 89120
877•8•POLOLU Fax: **702•262•6894**
Email: www@pololu.com
Website: www.pololu.com/smc

PROTOTYPE SOLUTIONS

Quickturn Prototypes in as Little as 24 Hours

Sunstone Circuits, has enhanced their services by offering prototype delivery within 24 hours on all four-layer boards ordered within their PCBexpress® quickturn product line.

Sunstone's PCBexpress® quickturn with predefined capabilities offers the best value for up to 100 pieces per order. By using this PCB design service, customers can now shave two days from the standard three-day lead time.

Customers are increasingly using multilayer board designs during the initial prototyping phase to dramatically shorten design-time and save project dollars. This technique works particularly well when high-speed design work may be a factor in the actual schematic.

This improved lead-time also comes with enhanced DFM rules which reduces the risk of design mistakes. When a prototype is meant to prove a concept or demonstrate a solution, adding layers to the design isn't that risky; risk it mostly a factor when engineers design on the border of the fabricator's limitations. This service now allows designers the ability to optimize their time by saving the 'tight-routing' elements for when prototypes become production ready. This equates to fewer prototype spins which consequently means reduced costs on PCB components, assembly, and labor.

For further information, please contact:

Sunstone Circuits

Website: www.Sunstone.com

COMBAT ZONE

Featured This Month:

Features

22 *PARTS IS PARTS:
Identifying Unknown
Brushed DC Motors*

by Kevin M. Berry

23 *Affordable 2.4 GHz*
by Pete Smith

27 *The Safe Use of Lithium
Polymer or Lithium-Ion
Batteries in a Combat
Robot*

by Steven Kirk Nelson

30 *RioBotz Combot Tutorial
Summarized – DC Motors*
Summarized by Kevin M. Berry

35 *Hello Bradley – Why
Gambling Doesn't Pay*
by Bradley Hanstack

36 *Melty Brains Cartoon*
by Sean Canfield and Kevin Berry

Events

35 *Oct 18th – Nov 10th 2010
Results and Jan/Feb 2011
Upcoming Events*

PARTS IS PARTS

Identifying Unknown Brushed DC Motors

● by Kevin M. Berry

This month's column is blatantly stolen (Editor's note: "researched," not stolen!) from Professor Marco Antonio Meggiolaro's popular book, the *RioBotz Combot Tutorial*. Marco's book is available free for download at www.riobotz.com.br/en/tutorial.html.

If you bought your motor from a junkyard or found it forgotten somewhere in your laboratory and you don't have any clue about its characteristics, you can follow these steps:

- Seek any identification on the motor, and look for its datasheet over the Internet.
- Make sure it is a DC motor. If there are only two wires connecting it, there is a good chance it is DC. Otherwise, it could be an AC, brushless, or step motor.
- Measure the electrical resistance between the terminals, obtaining R_{motor} .
- Apply increasingly higher voltages, such as 6V, 9V, 12V, 18V, and 24V. Wait for a few minutes at each level while checking if the motor warms up significantly. If it gets very hot even without loads, you're probably over the nominal voltage, so reduce its value.
- Most high quality motors can work without problems during a three minute match with twice their nominal voltage; this is a technique used in combat (such as the 48V Etek powered at 96V). The 24V Magmotors are exceptions. They are already optimized for this voltage, tolerating at most 36V. Even so, the current should be limited in this case.
- Once you've chosen the



FIGURE 1

working voltage V_{input} , connect the motor (without loads on its shaft) to the appropriate battery — the same that will be used in combat — and measure $I_{\text{no_load}}$. Note that the value of $I_{\text{no_load}}$ does not depend much on V_{input} . However, it is always a good idea to measure it at the working voltage. If you have an optical tachometer (which uses strobe lights, such as the one in **Figure 1**), you can also measure the maximum no-load motor speed $\omega_{\text{no_load}}$. A cheaper option is to attach a small spool to the motor shaft, and to count how long it takes for it to roll up. For instance, take 0 meters or 30 feet of nylon thread. The angular speed in rad/s will be the length of the thread divided by the radius of the spool; all this is divided by the measured time (the thread needs to be thin, so that when it's rolled up around the spool the effective radius doesn't vary significantly).

- Attach the motor shaft to a vise grip, hold both the motor and the vise grip well, and connect the battery. Be careful because the torque can be large. The measured current will be I_{stall} , associated with the circuit resistance $R_{\text{system}} = R_{\text{battery}} + R_{\text{motor}}$, so $I_{\text{stall}} = V_{\text{input}} / R_{\text{system}}$; then, calculate $R_{\text{battery}} = (V_{\text{input}} / I_{\text{stall}}) - R_{\text{motor}}$. Do not leave the motor stalled for a long time; it

will overheat and possibly get damaged. Also, take care not to dent the motor body while holding it (for instance, with a C-clamp) as shown in

Figure 2.

- Repeat the procedure above, but supporting one end of the vise grip by a scale or spring dynamometer (with the vise grip in the horizontal position; see **Figure 2**). Then, measure the difference between the weights with the motor stalled and with it turned off, and multiply this value by the lever arm of the vise grip to obtain the maximum torque of the motor, τ_{stall} . For instance, if the scale reads 0.1 kg with the motor turned off (because of the vise grip weight) and 0.8 kg when it is stalled, and the lever arm (distance between the axis of the motor shaft and the point in the vise grip attached to the scale) is 150 mm, then $\tau_{\text{stall}} = (0.8\text{kg} - 0.1\text{kg}) \times 9.81\text{m/s}^2 \times 0.150\text{m} = 1.03\text{N}\cdot\text{m}$.
- Because $\tau_{\text{stall}} = K_t \times (I_{\text{stall}} - I_{\text{no_load}})$, you can obtain the motor torque constant by calculating $K_t = \tau_{\text{stall}} / (I_{\text{stall}} - I_{\text{no_load}})$.

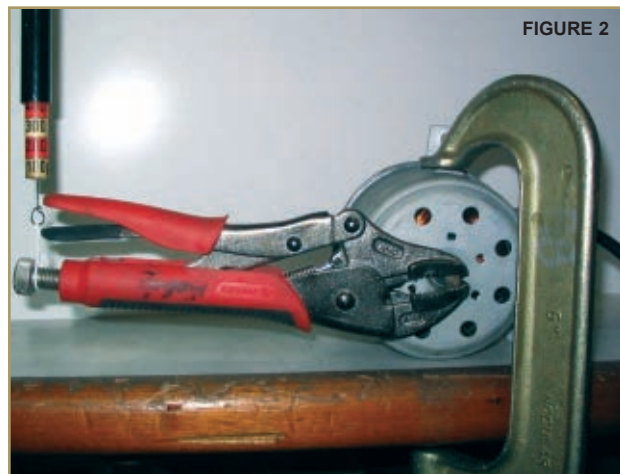


FIGURE 2

- Alternatively, if you were able to measure $\omega_{\text{no_load}}$ with a tachometer or spool, then you can calculate the motor speed constant using $K_v = \omega_{\text{no_load}} / (V_{\text{input}} - R_{\text{system}} \times I_{\text{no_load}})$. Check if the product $K_t \times K_v$ is indeed equal to 1, representing K_t in $\text{N}\cdot\text{m}/\text{A}$ and K_v in $(\text{rad/s})/\text{V}$. This is a redundancy check that reduces the measurement errors. If you weren't able to measure $\omega_{\text{no_load}}$, there is no problem. Simply calculate $K_v = 1 / K_t$, taking care with the physical units.
- Finally, once you have the values of V_{input} , K_t (and/or K_v), R_{system} , and $I_{\text{no_load}}$, you can obtain all other parameters associated with your motor + battery system using the previously presented equations (don't forget to later add the resistance of the electronics, as well). **SV**

Affordable 2.4 GHz

● by Pete Smith

The principal radio frequency for many years in USA combat robotics was 75 MHz PCM. This worked well enough, but the transmitters (TX) and receivers (RX) for PCM were both expensive, and the receivers were both bulky and not particularly reliable under combat conditions. The main problem, however, was the limited

number of channels available and controlling the use of those channels in a major event.

The arrival of the Spektrum DX6 changed all that. It uses 2.4 GHz and each radio "binds" to and can control only one RX at a time without any possibility that it can interfere with (or be interfered by) another transmitter. This removed —

in a stroke — many of the radio concerns at events. Organizers no longer had to worry about competitors interfering with each other or with other RC sets being used nearby, affecting safe control of the robots.

Competitors also no longer had to be concerned about being on the same channel as their opponent and perhaps being forced to make last

minute receiver crystal changes. They also don't have to worry about getting the appropriate frequency clip to be able to test their bot.

As an added bonus, the 2.4 GHz receivers were smaller, cheaper, and more reliable. Radio reception in the bot was also vastly improved and less susceptible to interference from the rest of the bot's electronics.

Events now commonly specify that 2.4 GHz must be used, so many competitors face the expenditure of about \$200 to get a new transmitter and receiver. However, a new range of very cheap 2.4 GHz equipment is now available and these bring the new technology within almost anyone's reach.

The radios are sold under a number of brands and with varying complexity, but the one I will be describing is probably one of the cheapest to buy that has all the functions that one would normally use in combat.

The HobbyKing HK-T6A (www.hobbyking.com) is a six channel, PC programmable set (**Figure 1**) and is usually sold for around \$25 plus shipping. The incredibly low price does come with a couple of drawbacks. The first is that it does not have a rechargeable battery or charger, so you'll need to buy eight AA alkaline batteries for it. The second problem is that the only way to program the radio is using

your PC via a special USB cable (an extra \$3). Finally, the radio comes without a manual so you need to go to the Internet to download this and a driver and setup program. These are all available for download from the Team Rolling Thunder website at www.teamrollingthunder.com.

The receiver comes in two sections; the second smaller part is a second antenna. I have found reception to be fine without this second antenna and so I remove it by unplugging it from the main receiver body.

Binding the RX

The first task after adding the eight AA cells to the radio is to bind the receiver to the transmitter. This means that this receiver will only work with this transmitter until such time as it is rebound to another transmitter. (You can, however, bind multiple receivers to any one transmitter.)

The procedure is as follows:

1. First, make sure the TX is switched off.
2. Insert the provided binding plug and link into the "BAT" pins and an RX battery (or power from your bot's BEC) to the "CH1" pins on the RX (**Figure 2**). Two LEDs should start flashing on the RX indicating it is ready to bind.
3. Press and hold the "bind range

test" button on the bottom left corner of the TX, then switch on the TX (on/off switch is on the lower right).

4. The LEDs on the RX should stop flashing after about 10 seconds indicating it has bound to the TX.
5. Release the "bind range test" button on the TX.
6. Remove the binding lead from the RX.
7. You can now test the TX by plugging in some spare servos to the RX and seeing if they respond to the TX stick movements. Be aware that channel allocations and responses may not be what you expect so do NOT connect the RX to your bot's weapon or drive motors at this time.

Loading the Device Driver

The second task is to load the device driver required for the USB cable. The required installation program can be found on my website at www.teamrollingthunder.com or from www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx. The drivers installed without issue on my computer (running Windows XP).

T6config Programming Software

The T6config program is used to



FIGURE 1



FIGURE 2

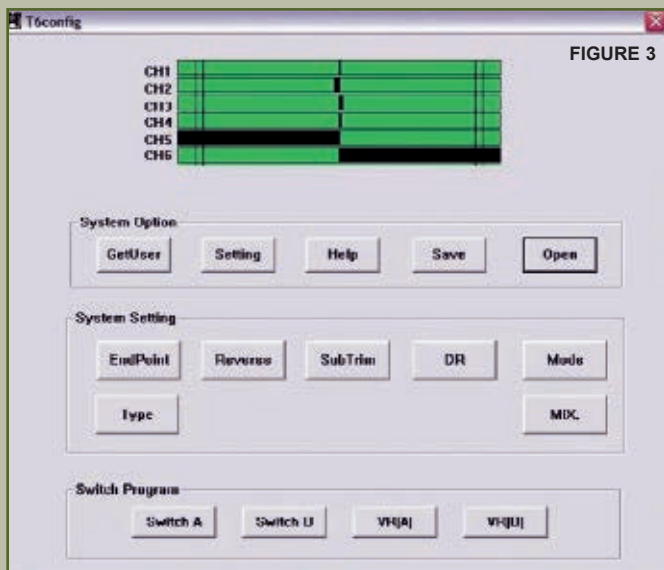


FIGURE 3

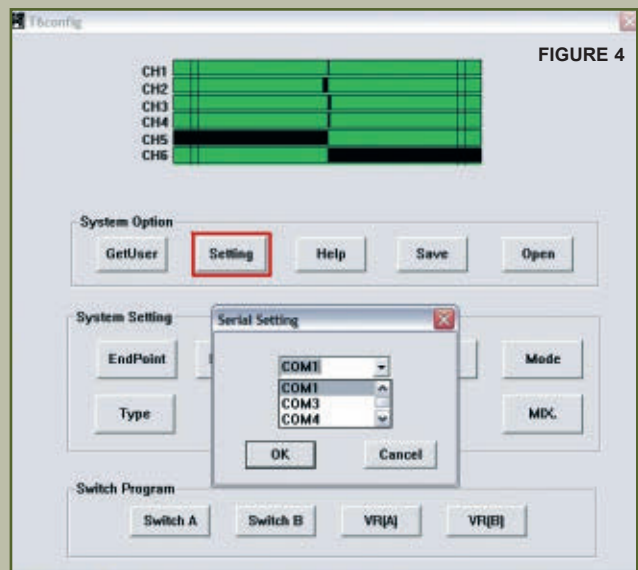


FIGURE 4

change the various settings on the TX. First, you must install the program t6config.exe. (Again, you can download that from my website.)

The install program must use a Chinese character set my computer doesn't have since it displays some unintelligible icons, but the program installs without needing user intervention at that point, so this doesn't present a problem. A T6config icon is added to your desktop.

Linking Your TX to Your PC

1. Power-up your computer.
2. Switch on your TX.
3. Plug in the round plug on the USB cable into the socket on the back of the TX.
4. Plug the other end of the USB cable into a spare USB port on your computer.
5. Launch the T6Config program on your computer. You should get a screen similar to that in **Figure 3**.
6. Click on the button marked "Setting" and a small window will pop up as shown in **Figure 4**.
7. Choose a COM port and click "OK."
8. If it is the right port when you move the sticks on the TX, the green bars will move in response.

If they don't, then repeat step 7 and choose another COM port until they do.

This is where I ran into a problem. No matter what COM port I tried, the bars would not move. I tried a different TX (I had purchased four) and a different USB cable (I had a couple), and finally tried it out on two other computers to no avail.

I researched online but could not find anyone who had the same problem. Finally, in desperation, I tried plugging it into

the powered USB hub I used on one of the computers and it worked immediately! I suspect that the power available from some USB ports is insufficient to correctly power the cable link, and the powered USB hub had a higher voltage or current capacity and allowed correct operation. The USB hub I used is shown in **Figure 5**. It came with a USB connection cable and a small transformer to power it. You can get hubs that are not separately powered but I doubt they would work.

FIGURE 5



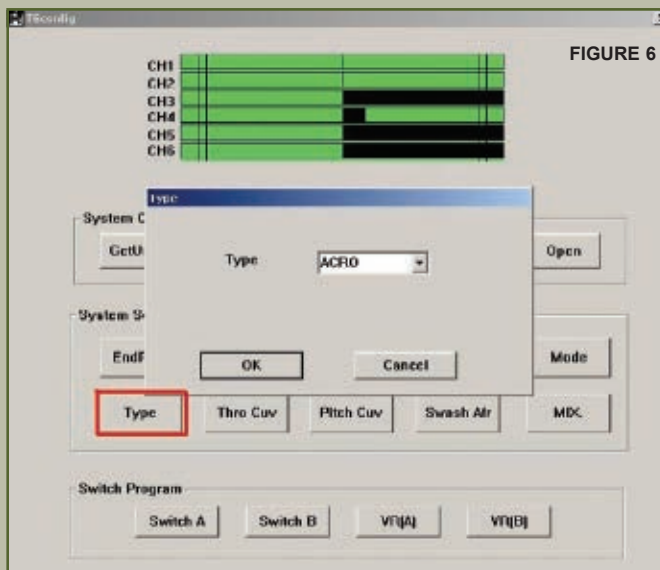


FIGURE 6

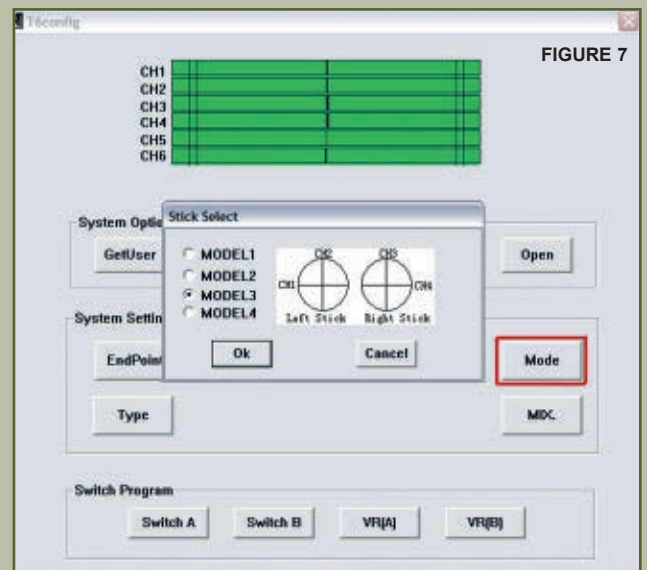


FIGURE 7

Programming Your TX

1. Click on the "GetUser" button in T6config. This uploads the default settings from the TX. Now, click on the "save" button to save the default settings onto your PC. Save the file somewhere that you will be able to find it again.
2. Click on the "Type" Button as in **Figure 6** and make sure "ACRO" is selected rather than one of the Helicopter settings.
3. In order to get the right hand stick on the usual channels (three and four), you click on the "Mode" button as in **Figure 7** and choose "Model 3" from the

pop-up menu. This still leaves Throttle on Channel 2 and rudder on Channel 1, but it's closer to the usual setup on a Spektrum or Futaba TX.

4. You can reverse any of the channels by selecting "reverse" (**Figure 8**) and selecting or deselecting as required.
5. Dual rating is a little more complex. First, you must select what channels you want to dual rate by clicking on the DR button (**Figure 9**) and changing the values for DR ON and DR OFF; 100 and 18 worked well on Channel 4 in my bot hockey robots to calm down the steering

response. Set all other channels (those that do not require dual rating) to 100 and 100. Now, you have to select which switch is used to operate this function. Click on the "Switch A" button (**Figure 10**) and select DR. This will set the switch at the top right of the TX to switch between the rates.

6. Mixing for Tank steer (if this is not done by the Drive ESC) can set up through use of the "Mix" button (**Figure 11**). The values shown worked well for my bots.
7. Similarly, the servo end points and sub trims can be set using the buttons as marked.

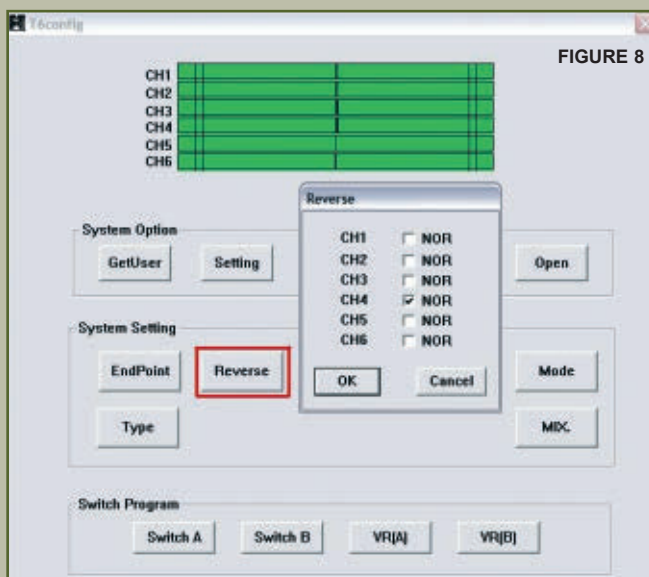


FIGURE 8

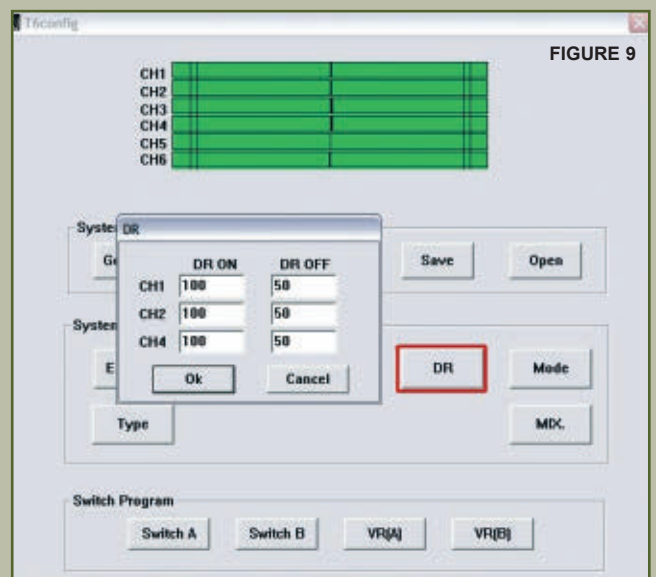


FIGURE 9

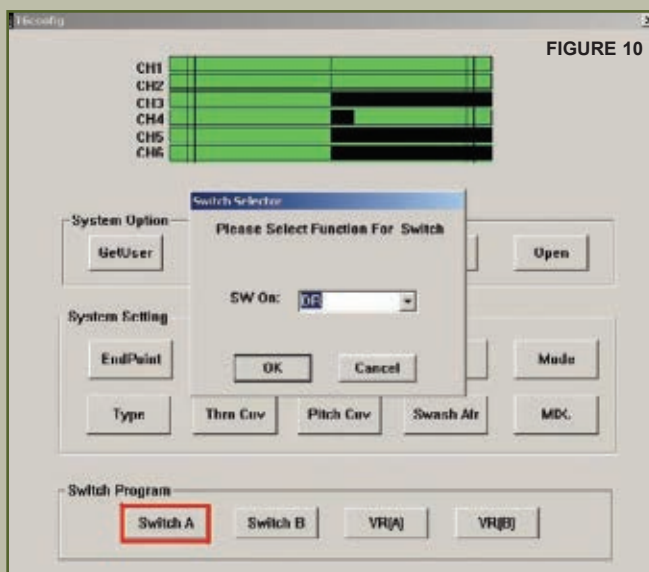


FIGURE 10

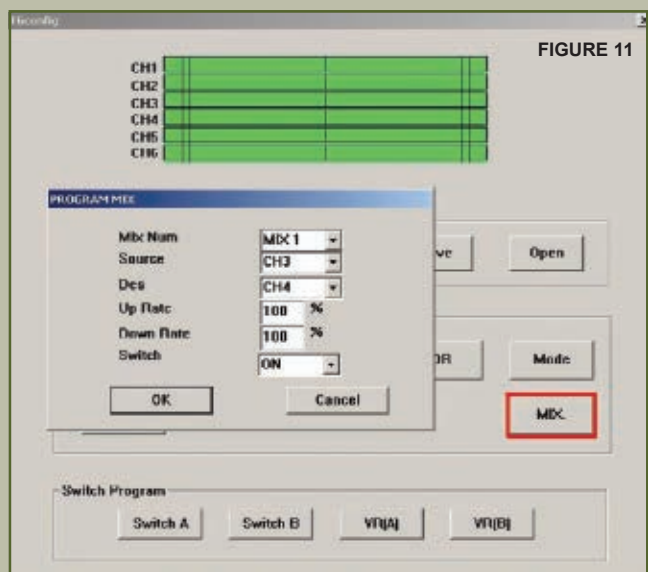


FIGURE 11

Save this file under a new file name in the same place as you saved the default settings. Click on "GetUser" and check that it has saved to the TX, as well. If it has not, then open the saved file using the "Open" button. The settings should now be downloaded to the TX. A file — Servomix — is available on my website that has the settings I use with two Banebots ESCs in my hockey bots. You can download this and then open it using TConfig to give you a start in setting up your own bot.

Switch off the TX and unplug the USB cable. You can now plug

the drive motor ESCs into RX Channels 3 and 4 (ensure your weapon blade or drive belt is removed for initial testing). Power-up your bot and see how the bot responds to the controls. You may have to swap the ESCs around and swap the drive motor leads to get everything to run in the correct direction. If necessary, reattach the TX to the computer as before and change your configuration as required (reverse servos, tune dual rate settings, etc.) until you get the bot running just how you want it to.

Remember also that unless you

have a laptop, you cannot change the TX settings at a competition, so make sure it's right before you go! Also, all your bots must use the same settings.

If you plan to use the TX a lot, then it's probably worthwhile to get a rechargeable TX pack. However, I could not find a battery/charger combo designed for this radio.

While setting up these cheaper 2.4 GHz radios is a little more complex than a standard computer radio like the Spektrum DX6i, they do work well and the price makes them a lot more affordable. **SV**

The Safe Use of Lithium Polymer or Lithium-Ion Batteries in a Combat Robot

● by Steven Kirk Nelson

I recently got back from COMBOTS 15 where I was the arena wrangler. Basically, my job was to provide safety and also train several new

wranglers in the procedures used in running an event. A wrangler's job is not an easy one. You have to control the loading and unloading of the

robots into the arena, control the power-up and power-down sequence of the robots, plus — when needed — enter the arena and



unstuck the active robots from the arena wall or each other. Wranglers also put out actual fires that can be electrical, chemical, or other combustible materials in nature. Over the years, many fires have occurred, usually because of electrical shorts or spilled fuel. They have been pretty easy to suppress using CO₂ (Carbon Dioxide) or dry chemical ABC fire extinguishers.

Most of the time, electrical fires can be put out by simply turning off the main power switch on the robot to stop the flow of current from the batteries. In the past, the battery technology used was mainly Sealed Lead Acid, Nickel Cadmium, or Nickel Metal Hydride. These types of batteries tend to fail in relatively non-spectacular ways. They tend to short themselves out quickly or blow open their internal connections, then die. They are easily removed, extinguished, and properly discarded. So, it's pretty rare that we ever even use a fire extinguisher.

Recently, builders have been using a new, more powerful battery type based on Lithium Polymer or Lithium-Ion chemistries. Originally, these batteries were used in mainly small combat robots (1-30 lb), mostly because of their costs. Now that has changed, and the larger robots (60-340 lb) are using them and in much larger quantities. In

small robots, there may be packs providing 1-5 amp-hours capacity, but in larger robots there may be over 20 amp-hours of capacity using multiple packs in parallel.

Lithium Polymer battery technology is really amazing. It provides a very high energy density by weight compared to all other battery types used in combat robots. One of the reasons for this is that unlike most cells, they do not have a strong metal or plastic case to protect the cells from external damage. Because they are both lightweight and less bulky, they make the builder's job easier when looking for ways to fit them in the robot. Of course, their construction also makes them more fragile than the other types of cells.

Lithium Polymers have some other characteristics that differ from the older battery types. They are much more sensitive to voltage imbalances between individual cells, voltage differences between cell packs, and excess current loads during discharge. They also tend to run hotter under load and when charging. In combat, robotics machines are constantly subjected to very high impact forces and severe mechanical damage. It's not uncommon to see the armor torn off the robots and internal parts exposed or torn up including the

battery packs.

Lithium Polymer batteries cannot handle physical damage and will likely catch fire or produce huge volumes of smoke once their lightweight packaging is breached. Once they become damaged, they may continue to burn, smoke, or even re-ignite until their internal energy is depleted. It should also be mentioned that excessively high current draws caused by stalled electric motors, pinched wire, or other types of short circuits may also cause this problem.

Sewer Snake vs. Death and Taxes Fight

During a 220 lb Heavyweight fight at COMBOTS 5, there was a match between a robot called Sewer Snake and a multibot (a team of two smaller robots) called Death and Taxes. The multibot consisted of a large (approximately 120 lb) robot called Death with a drum spinner weapon and a smaller, very formidable 50 lb robot called Taxes (using a titanium wedge shaped body). At about 30 seconds into the three minute match, Taxes was hit by his teammate's spinning weapon on its right rear corner; its titanium armor was breached. One of the 5.5 amp-hour Li-poly packs was physically damaged and the battery started smoking. Usually, this is a death sign for a combat robot, but Taxes kept fighting as the smoke began to fill the arena.

The amazing power demonstrated from this robot's six horsepower drive train and its battery technology was mesmerizing and entertaining. Eventually, the battery pack actually caught fire, and yet Taxes kept fighting — even carrying a 220 lb robot on its back. The fire then subsided for several seconds but the smoke kept pouring out of the robot.

In about another minute, the robot got stuck under the arena wall and then it caught fire again. The builder asked for an "unstuck." The

fight was temporarily stopped and this left me and the event organizers with a problem. Should we let a wrangler enter the arena to put out the fire and unstick the robot while the entire arena was filled with thick white smoke? We eventually did allow a wrangler to enter the smoke filled arena and hit Taxes with a blast from a CO₂ fire extinguisher. A couple of minutes later, I also briefly entered the arena and unstuck the robot, and the fight resumed for another 30 seconds or so. However, Taxes couldn't move at this point.

After the match was ended, we opened the arena doors to help vent the smoke into the building. (It should be noted that both the arena and the building had ventilation fans.) After several minutes, the robots were powered down and Taxes was removed from the building to an area where we had two buckets of dry sand to smother the batteries once they were removed from the robot. The batteries kept smoking and still were a fire hazard for almost an hour. This also stopped the event for over 30 minutes while the smoke cleared.

In a discussion with the builders of Taxes, it was determined that probably the physical damage to the first pack caused the battery failure and its open flames eventually burned away the added fire resistant foam protecting the pack, and then the one next to it. This caused the second pack's packaging to fail and it ignited and burned, as well. It was also mentioned that the robot was not designed for use with Li-poly batteries but they were added for this event. This started some discussion on the Robot Fighting League forum about better ways of building with Lithium Polymer or even Lithium-Ion battery types.

Some Ideas and Tips from the Builders of Death and Taxes

(This information was graciously

provided by Billy Moon and Dick Stuplich.)

- Always use LipoSacks in the pits and for shipping batteries when they are not in a Ti container.
- Always use battery boxes for the Lithium Polymer packs. When building those boxes, use overlapping sides (no butt seams). Allow for a 'chimney' through the middle of the packs and plan for the fire/smoke to exit there. Anything above/below that chimney will get damaged.
- Always shock-mount the battery box rather than hard-mounting it. If you have a multi-cell pack, then make sure to orient the packs so that the most likely impact blows are along the 'length' of the pack.
- Tape down and insulate open terminals on the balance plugs so that they don't get cut or shorted.
- Use fairly stiff foam to keep the packs from shifting around in the robot.
- Design so as not to draw more than 50% of the capacity of the batteries and to never exceed the continuous draw capacity of the packs. For example, in Taxes, there were two 5,000 mAh packs, 125 amps continuous, 250 amps burst. The speed controller was set for 80 amps per channel, so its maximum continuous draw was 160 amps. This required two packs so that the continuous draw capability was 250 amps. Don't be tempted to violate this rule. LiPoly's are small and light so you can give them enough juice. In Death, there were four of those packs. There were also four speed controllers: two sets for a maximum of 80 amps and two sets for a maximum of 160 amps. The robot needed 480 amps continuous, so four packs were put in to give 500 amps continuous.
- The packs never come out of battle more than 30% discharged and are barely warm. For a long battle, it typically takes 1.5-1.8 Ah

to recharge the packs. Fan-cool the batteries before charging.

- Never charge at more than .75C (3.5 amps maximum), even though they are rated at 3C.
- You should also re-balance the packs at the end of each combat day.
- Li-poly battery fires can exceed 760 deg. C (1,400 deg. F) which is more than enough to ignite most combustible materials.
- Another thought from a different builder (Jerome Miles) was to mount the batteries inside of a LipoSack inside of the robot to prevent the open flames from causing secondary fires, or propagation and failure from one pack to another inside of the robot. The LipoSacks can provide good fire containment as long as they are not torn or damaged.

Fighting the Fires and then Dealing with the Batteries

Li-poly fires are a bit different than other battery fires. For the most part, the open flame can be extinguished with a CO₂ or dry chemical ABC extinguisher, although is is possible and likely that the pack may re-ignite because of its internal energy. There are commercial fire extinguishers available just for this type of battery available using Ansul Lith-X. There is also a copper powder. It's been proven that just covering the batteries in plain dry sand works really well.

It is not recommended to pour water on the batteries if the case has been breached. This actually varies for different battery chemistries but as a rule of thumb, water is not the best option. Water coming in contact with the battery electrolyte can produce hydrofluoric acid. Water coming in contact with the battery anode material can produce flammable hydrogen gas.

The smoke produced from these batteries is a health hazard. The

LINKS

Sewer Snake and Death and Taxes fight videos

www.youtube.com/watch?v=IUBBOPBbjpk&sns=em

Another camera view of the fight and fire

www.youtube.com/watch?v=amvqgyqtr74

RC Group's complete guide to Li-poly batteries

www.rcgroups.com/forums/showthread.php?t=209187

LipoSack

www.liposack.com/products.htm

Fastrax LiPo Charge Bag. This bag is made from Kevlar and Nomex

www.cmldistribution.co.uk/cml_product.php?productId=0000004476

Ansul Lith-X

www.ansul.com/en/support/search_combo.aspx?txtSearchPhrase=lith-x

smoke contains several toxins. It is recommended that protective clothing and a full-face SCBA breathing system be used by anyone fighting this type of fire. The smoke is one of the more difficult issues to control and it's a good idea for event organizers to provide a high flow ventilation

system for their arenas and buildings. They should not allow people to enter the arena until the smoke has been cleared. Due to the nature of this battery chemistry and the volume of smoke produced, this could take a long time. This is definitely something to consider and in some event venues, it may be difficult to achieve.

At Combots, We had a Simple Battery Fire Plan

During the safety meeting, the builders were told that they were responsible for removing the robot from the arena and the building when it was judged safe to do so. There was a clear path maintained from the arena to the outside doors of the building. The builders were shown the path to the exit. The builders were instructed to wear leather welding gloves if they had to handle the hot robot. Steel carts were provided either by the builders or the event to place robots on, so they could be quickly wheeled outside.

When possible, the event crew would put out the open fire with CO₂.

The builder would then remove

the robot from the building, then remove the batteries from the robot.

The builder would smother the batteries with the provided dry sand.

The Future

It should be noted that the use of other cells like the Lithium-Ion A123 brand have much less of a chance of fire or this smoke hazard. Of course, they are bulkier and a little heavier than many of the Li-poly packs. They may prove to be a better idea but not quite yet ... trial and terror testing is still the nature of the game.

As usual, combat robotics consistently pushes available technology to its limits and often way beyond; this is the nature of the sport. Hopefully, with a better understanding of this impressive and powerful battery technology, better building practices, and improved event safety measures we can continue to provide a great show and high standards of performance in this dangerous and yet safe sport. **SV**

*Photo by Jon Swenson.
See more of Jon's work at
www.sharkspage.com/?p=2668.*

RioBotz Combt Tutorial Summarized – DC Motors

● Original Text by Professor Marco Antonio Meggiolaro; Summarized by Kevin M. Berry

Editor's note: Professor Marco Antonio Meggiolaro, of the Pontifical Catholic University of Rio de Janeiro, Brazil, has translated his popular book, the RioBotz Combots Tutorial, into English. In previous editions of the Combat Zone, SERVO has summarized many portions of the tutorial. In this article, we present a much simplified version of the "Brushed DC Motors" section of

Chapter 5 – a major treatise on bot motors. Marco's book is available free for download at www.riobotz.com.br/en/tutorial.html, and for hard copy purchase (at no profit to Marco) on Amazon, published by CreateSpace. All information here is provided courtesy of Professor Meggiolaro and RioBotz. In reviewing this article, Prof. Marco took the

opportunity to update some information that is now obsolete in the published version.

Brushed DC Motor Overview

Motors are probably the combat robot's most important component. They can be powered electrically, pneumatically,

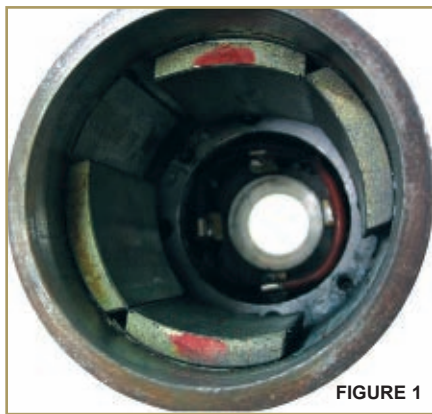


FIGURE 1

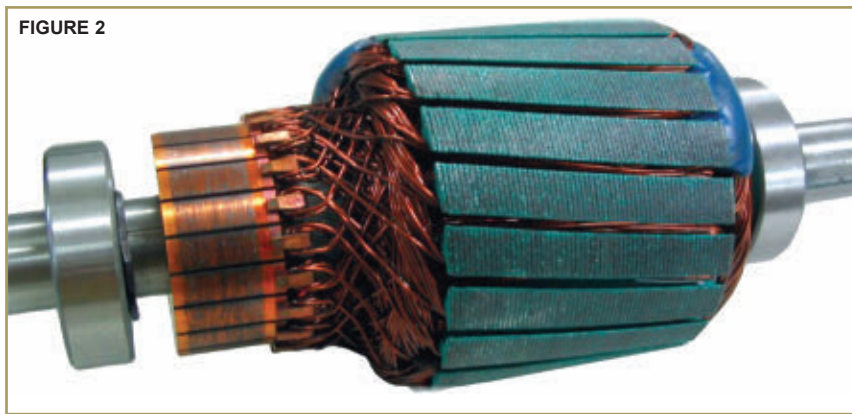


FIGURE 2

hydraulically, or using fuels such as gasoline. One of the most used types is the brushed direct current (DC) electrical motor because it can reach high torques, it is easily powered by batteries, its speed control is relatively simple, and its spinning direction is easily reversed. They are also a good choice because they're not as expensive as they used to be. There are other types of electrical motors, but not all of them are used in combat. For instance, stepper motors have — in general — a relatively low torque compared to their own weight. The speed of AC motors is more difficult to control when powered by batteries which can only provide direct current.

The three main types of brushed DC motors are the permanent magnet (PM), shunt (parallel), and series. The series type motors are the ones used as starter motors; they have high initial torque and high maximum speed. If there is no load on their shaft, starter motors would accelerate more and more until they would self-destruct which is why they're dangerous. In a few competitions, they can be forbidden for that reason. They are rarely used in the robot's drivetrain because it is not easy to reverse their movement; however, they are a good choice for powerful weapons that spin in only one direction.

The PM DC motors and the shunt type have similar behaviors, which are quite different from the starter motors. The PM ones are

the most used, not only in the drive system but also to power weapons. They have fixed permanent magnets attached to their body (as shown in **Figure 1**) which forms the stator (the part that does not rotate), and a rotor that has several windings (**Figure 2**). These windings generate a magnetic field that — together with the field of the PM — generates torque in the rotor. To obtain an approximately constant torque output, the winding contacts should be continually commutated which is done through the commutator on the rotor and the stator brushes (pictured in **Figure 3**). Electrically, a DC motor can be modeled as a resistance, an inductance, and a power source, connected in series. The behavior as a power source is due to the counter electromotive force which is directly proportional to the motor speed. The choice of the best brushed DC motor depends on several parameters, modeled next.

To discover the behavior of a brushed DC motor (permanent magnet or shunt type), it is necessary to know some parameters:

- τ — Applied torque at a given moment.
 - ω — Angular speed of the rotor.
 - P_{output} — Mechanical power output.
 - η — Efficiency = $P_{\text{output}}/P_{\text{input}}$ which results in a number between 0 and 1. In an ideal motor (which doesn't exist in practice), there would be no electrical resistance and no mechanical friction losses; in that case, $\eta = 1$ (100% efficiency).
- The curves in **Figure 4** show the drawn current I (I_{input}), angular speed ω , output power P_{output} , and efficiency η as a function of the torque τ applied to the motor by a load such as a wheel or spinning weapon.
- (Editor's note: A section showing the use of many interesting but slightly scary equations to derive the curve below and its results were omitted from this summary.)*
- $I_{\text{no_load}}$ — Electric current drawn by the motor to spin without any load on its shaft.
 - I_{input} — Electric current that the motor is drawing.
 - I_{stall} — Electric current drawn by the motor when so much load is applied it can't turn at all.

FIGURE 3



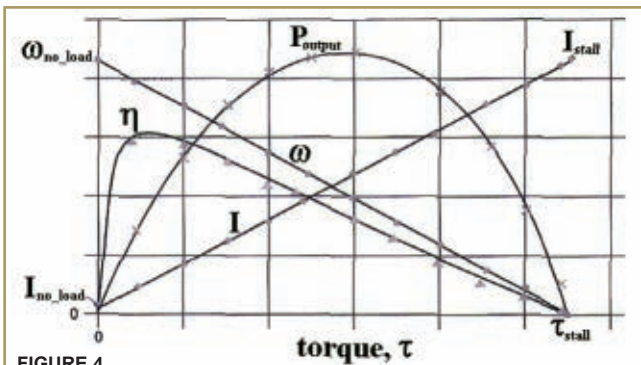


FIGURE 4

The plot in **Figure 4** shows that:

- The maximum speed ω_{no_load} happens when the motor shaft is free of external loads, with $\tau = 0$.
- The maximum current I_{stall} happens when the motor is stalled, with speed $\omega = 0$, so at I_{stall} the motor is generating the maximum possible torque.
- The maximum value of the mechanical power P_{output} happens when ω is approximately equal to half of ω_{no_load} .
- The highest efficiency happens in general between 80% and 90% of ω_{no_load} .

Example: Magmotor S28-150

The guide calculates the values (shown in **Figure 5**) for the popular

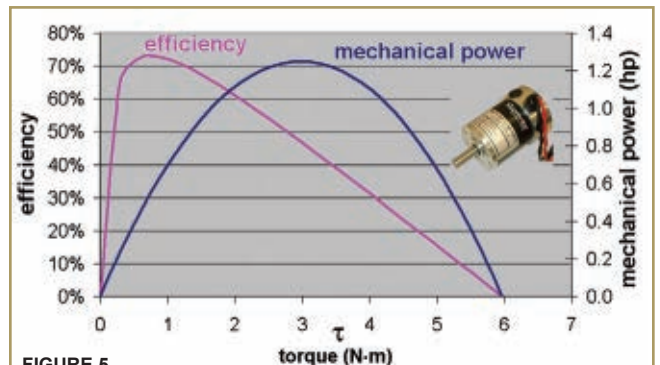


FIGURE 5

Magmotor S28-150 connected to one NiCd 24V battery pack. The guide's "teaching point" — aside from providing sample calculations — is twofold.

First, the calculated maximum input power — 5.2 HP (at stall), — does not mean you'd actually get that much horsepower. All this power is wasted when the motor is stalled; converted into heat by the system resistance. The maximum mechanical power is actually calculated to be 1.25 HP. Notice that the manufacturer says that the maximum power is 3 HP for that motor which you would only get if the battery and electronic system resistances were zero, leaving only the motor resistance R_{motor} (not a real world case.)

The second point is that (as it can be seen in the graph) the maximum mechanical power happens at speeds that are not necessarily efficient.

item). Their actual values in a battery/controller/wire/motor system would not be as good. (Note: K_t and K_v — not used elsewhere in this summary — are the motor torque and motor speed constants.)

The Bosch GPA and GPB shown in the **table** have been extensively used in Brazil to drive middleweights. However, they have a low ratio between maximum power and their own weight. In addition, the GPA generates a lot of noise which can reduce the range of 75 MHz radio control systems. This problem can be minimized using capacitors between the motor brushes, or switching to (for instance) 2.4 GHz radio systems.

The DeWalt 18V motor with gearbox is a good choice for the drive system; we've used it in our middleweight Cyclone. It has an excellent power-to-weight ratio. Its main disadvantages are that it is not easy to mount to the robot structure, the gearbox casing is made out of plastic, and its resulting length including gearbox ends up very high to fit inside compact robots. Note that some older discontinued DeWalt cordless drills had other disadvantages, using Mabuchi motors instead of the higher quality DeWalt ones, and using a few plastic gears among the metal ones in their gearbox.

The NPC T64 already includes a gearbox with typically a 20:1 reduction. The data in the **table** already include the power loss and



FIGURE 6

Typical Brushed DC Motors

The Magmotor example can be repeated for several other motors. **Table 1** shows a few of the most used motors in combat robots and their main parameters. Several parameters are based only on motor specs (as a standalone

weight increase due to the gearbox which explains the relatively low power-to-weight ratio. But, even disregarding that, the performance of this motor is still not too high. The reason many builders use it is due to its convenience, it is easily mounted to the robot, and it is one of the few high power DC motors that comes with a built-in gearbox. Care should be taken with the NPC T64 gears (shown in **Figure 6**, with red grease). They might break under severe impacts if used to power weapons. As recommended by the manufacturer, only use them as drive motors.

An excellent motor for driving middleweights is the Magmotor S28-150 (a.k.a., Ampflow A28-150). It is used in our robots Titan and Touro. A good weapon motor for a middleweight would be the Magmotor S28-400 (a.k.a., Ampflow A28-400) with higher torque and power, which we use to power Touro's drum. Using a single S28-150 to power the weapon of a middleweight is not a good idea. There's a good chance that it will overheat.

Because of that, to spin the bar of our middleweight Titan, we use two Magmotor S28-150s mechanically connected in parallel by acting on the same gear of the weapon shaft.

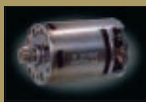
The D-Pack motor is a good candidate to replace the Magmotors, besides being much cheaper. However, its electrical resistance is so low that it almost shorts the batteries and electronics. Because of that, its current must be limited if used with speed controllers. Otherwise, there's a good chance of damaging the electronics, especially since this motor is usually overvoltage to 24V

instead of powered by its nominal 12V. If used with solenoids to power weapons, make sure that they can take the high currents involved. This motor is difficult to find even in the US.

The Etek motor is really impressive. It may deliver up to 15 HP (1 HP = 746W), and it can deliver high torque and high speed at the same time. It is a little too heavy for a middleweight. We ended up using it in our spinner Ciclon but we had to power it at only 24V because the additional battery packs that would be needed to get to 48V would make the robot go over its 120 lb weight limit. The super-heavyweight shell spinner Super Megabyte only needs one of these motors (powered at 48V) to spin up its heavy shell. A few daring builders

have overvoltage to 96V, but current limiting is highly recommended.

A few DC motors allow the permanent magnets fixed in their body to be mounted with an angular offset with respect to their brush housings (typically about 10 to 20 degrees, it depends on the motor) which allows you to adjust their phase timing. If the motor is used in the robot drive system, it should have neutral timing. In other words, it should spin with the same speed in both directions, helping a tank steering robot to move straight. If it is used to power a weapon that only spins in one direction, you can advance the timing to typically get a few hundred extra RPM. (On the other hand, in the other direction the

TABLE 1				
Name	Bosch GPA	Bosch GPB	D-Pack	DeWalt 18V
Voltage (V)	24	12	12 (nominal)	24
P _{output_max} (W)	1,175	282	3,561	946
Weight (lb)	8.4	3.3	7.7	1.0
Power/Weight	140	85	462	946
I _{stall} /I _{no_load}	23	25	63	128
K _t (N×m/A)	0.061	0.042	0.020	0.0085
K _v (RPM/V)	167	229	485	1,100
R _{motor} (W)	0.13	0.121	0.00969	0.072
I _{no_load} (A)	8.0	3.9	19.6	2.6

				
Name	Etek	Maggmotor S28-150	Maggmotor S28-400	NPC T64 (w/gearbox)
Voltage (V)	48	24	24	24
P _{output_max} (W)	11,185	2,183	3,367	834
Weight (lb)	20.7	3.8	6.9	13.0
Power/Weight	540	574	488	64
I _{stall} /I _{no_load}	526	110	127	27
K _t (N×m/A)	0.13	0.03757	0.0464	0.86
K _v (RPM/V)	72	254	206	10
R _{motor} (W)	0.016	0.064	0.042	0.16
I _{no_load} (A)	5.7	3.4	4.5	5.5



FIGURE 7

motor speed would decrease.)

To advance the timing, loosen the motor screws that hold its body, power it without loading its shaft, and slightly rotate its body (where the permanent magnets are attached to) until the measured I_{no_load} current is maximum, and then fasten the body back in place. For neutral timing, rotate the body until I_{no_load} is identical when spinning in both directions.

Regarding hobbyweights (12 lbs, about 5.4kg), a few inexpensive gearmotor options for the drive system are the ones from the manufacturers Pittman and Buehler, which can be found in several junk yards. Our hobbyweight drumbot Tourinho originally used (in 2006) two Buehler gear motors (with 300 grams each, about 0.66 lb), and our hobbyweight wedge Puminha used four Pittmans (with 500 grams each, about 1.10 lb). We've bought used ones in Brazil for about US\$10 to US\$15 each (after bargaining). Most of them have nominal voltage at 12V. However, we've used them at 24V for three minute matches without overheating problems. Remember that by doubling the voltage, the power is multiplied by four.

The only problem is that the

small gears can break due to the higher torques at 24V — we've broken quite a few 12V Pittmans after abusing them in battle at 24V. The only way to know whether they'll take the overvolting is by testing them. It's also a good idea to always have spare motors.

There are much better gearmotor options for hobbyweights and even heavier robots than the ones from Pittman and Buehler, however, they usually need some modifications to get combat-ready. We've been using Integy Matrix Pro Lathe motors (Figure 7) for the drive system in our hobbyweights, I adapted to 36 mm BaneBots (www.banebots.com) gearboxes that were modified following Nick Martin's recommendations that were described in the March '08 edition of *SERVO Magazine*. Recently, we've upgraded the gearboxes to BaneBots' P60.

A good combination for the drive system of a featherweight is the discontinued 42 mm BaneBots gearbox (or its improved version Magnum 775), connected to a 775-sized motor. For Touro Feather's drivetrain, we've adapted 18V DeWalt motors to modified Magnum 775 gearboxes with great results. This overkill combination for

a featherweight can even drive a lightweight. Other lightweight drive options are 18V DeWalts connected to custom-made gearboxes, such as the DeWalt Powerdrive Kit or Team Whyachi's TWA69 gearbox with Astroflight 990 Cobalt motors which we use in Touro Light.

For middleweights, S28-150 Magmotors are usually a good choice for the drive system, connected (for instance) to Team Whyachi's famous TWM 3M gearbox (pictured in Figure 8). The S28-400 Magmotors are more appropriate for the drive system of heavyweights and super heavyweights, connected (for example) to the TWM 3M gearbox (used in Touro Maximus) or to the stronger TWM3.

A good option for the drive system of beetleweights used in Mini Touro is the Beetle B16 gearmotor (Figure 9), sold at The Robot Marketplace (www.robotmarketplace.com). For antweights and fairyweights, the Sanyo 50 micro geared motor (Figure 10) is a very popular choice.

There are several other good brushed DC motors besides the ones presented above, not only for the drive system, but also to power the weapon. Brushless motors (to be studied in a later article) have been successfully used as weapon motors in several weight classes. It is useful to do research on which motors have been successfully used in combat. Several motors can be found at The Robot Marketplace and much more information can be obtained in the RFL Forum (<http://forums.delphiforums.com/therfl>).

Summary

I've managed to grossly simplify this subject in this tutorial. Detailed calculations are available for those wanting to understand the math behind these seemingly simple (but actually amazingly complex) pieces of bot technology. **SV**



FIGURE 8



FIGURE 9



FIGURE 10

EVENTS

Completed and Upcoming Events

Completed Events for Oct 18th – Nov 10th

Robothon Robot Combat 2010 was presented by Western Allied Robotics in Seattle, WA on October 24th. There were 20 bots registered in this event.



Mecha-Mayhem 2010 was presented by the Chicago

Robotic Combat Association in Rosemont, IL on October 23rd and 24th. There were 23 bots registered in this event.



COMBOTS Cup V was presented by COMBOTS in San Mateo, CA, on October 23rd and 24th.



Upcoming Events for January – February 2011

BB 2011 Nationals will be presented by BattleBots at the Coconut Grove Expo Center in Miami, FL on February 23rd–27th. Go to www.battlebots.com for more information. **SV**



Hello Bradley – Why Gambling Doesn't Pay

● by Bradley Hanstack (www.TeamThinkTank.com); Photo by Wendy Maxham

Editor's note: One thing the general public might not know about combat bot-ka-teers is they have a terrific sense of humor. The nature of the sport also makes them very "accountable." In this episode of "Kids, please don't gamble," Bradley learns an important life lesson!

Oh dear, what did I get into? The truth is I did lose a bet, but the truth has been a bit skewed, so here is the real story of how I ended up in Hello Kitty PJs at COMBOTS.

In 2009, at the Battlebots Pro Championship up near San Francisco, CA I had made a bet with my teammate Ted Shimoda. The bet was if Judge Dave (Dave Calkins) would attend the event. I, of course, thought that he would never show up since Dave hosts his own events. Ted made the bet and the loser would have to wear Hello Kitty PJs to the next competition. After I lost, Ted changed the punishment to just

paying for a dinner.

The bet was over and I didn't have anything to worry about until I decided it would be funny to mention this bet to Dave over the forums. The next day, I had an email forwarded to me from Amazon about an XL Hello Kitty PJ purchase.

I was forced into wearing the PJs and was paraded around the arena in order to compete at COMBOTS Cup V with only my **Dice.com** fuzzy dice around my neck to mask the pain. Highlight of the week for some, dreaded memory for a select few ... well, just me. **SV**



Melty Brains

by Sean Canfield and Kevin Berry

Anybody remember the last time we actually held an event?



WELCOME!
RFL RULES CONVENTION 2019
EST. ATTENDANCE 15,234

It was right before we got on Judge Dave's last nerve!

Mandatory Weapons!
No More Boxbots!

Why are they called "robots" anyway?

I think I just heard Steve Judd sigh.

Can this be the year we finally define what a "walker" is?

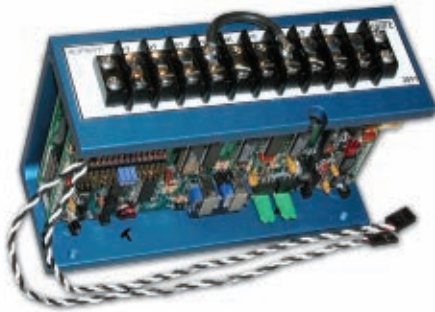
CAN SUMBUDY tel me how to bild A bOT?

Free The Wedges!

Wedges Suck!

You Suck!

STEER WINNING ROBOTS WITHOUT SERVOS!



Perform proportional speed, direction, and steering with only two Radio/Control channels for vehicles using two separate brush-type electric motors mounted right and left with our **mixing RDRF dual speed control**. Used in many successful competitive robots. Single joystick operation: up goes straight ahead, down is reverse. Pure right or left twirls vehicle as motors turn opposite directions. In between stick positions completely proportional. Plugs in like a servo to your Futaba, JR, Hitec, or similar radio. Compatible with gyro steering stabilization. Various volt and amp sizes available. The RDRF47E 55V 75A per motor unit pictured above.
www.vantec.com

VANTEC

Order at
(888) 929-5055

The Robot MarketPlace

Over 10,000 products available!

- Motors
- Batteries & Chargers
- Vex Robotics
- Carbon Fiber
- Wheels & Tires
- Speed Controllers
- R/C Systems
- Drive Components
- Hobby Supplies & Toys

(877) ROBOT 99
(877-762-6899)

Your One-Stop Robot Shop!
RobotMarketPlace.com

bots IN BRIEF

FINGERS GET THUMBS UP

Touch Bionics (www.touchbionics.com), developer of advanced upper-limb prosthetic technologies, has won a Best of What's New Award for 2010 from *Popular Science* for their ProDigits solution.

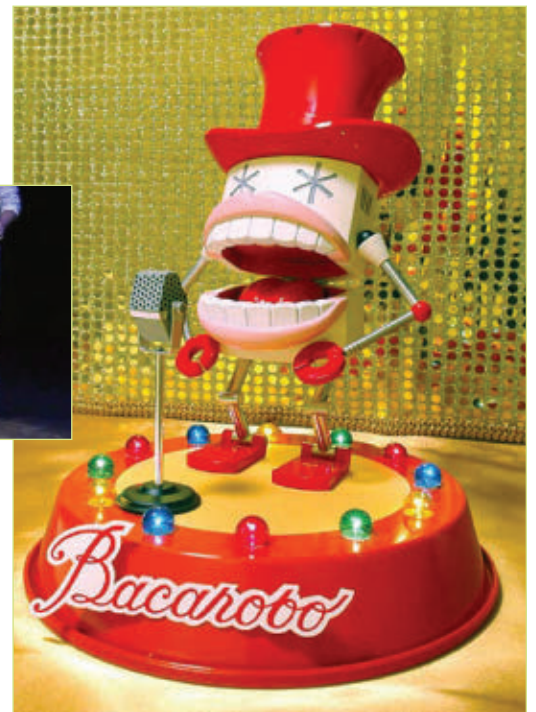
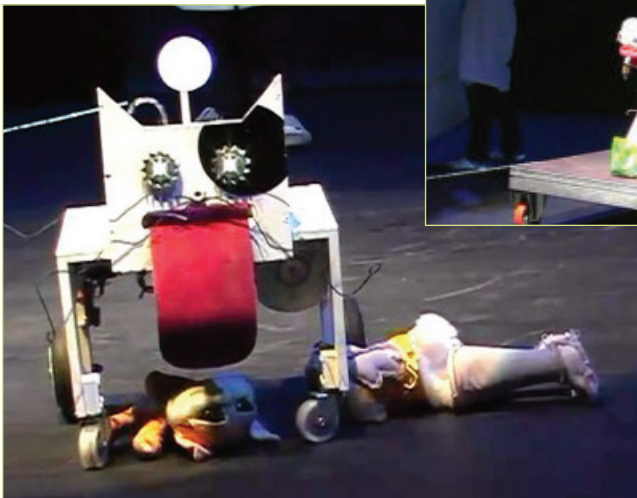
ProDigits is the world's first powered prosthetic solution for people with missing fingers. With ProDigits, Touch Bionics extended the technology innovation behind its groundbreaking i-LIMB hand to create a solution that brings life-changing technology to partial-hand patients. The global population that can benefit from the system is considerable — estimated at around 1.2 million worldwide — and before ProDigits such people had no powered prosthetic available to them.

"With ProDigits, I gain more independence. I can pick something up and walk out to the car with it, rather than have to put it in a bag," says ProDigits wearer, Eric Jones (pictured here). "Most importantly, I'm able to take care of my kids — play games with my kids, take them to school, make dinner. ProDigits helps with all that."



CHUCKLING FOR DOLLARS

We all love robots, especially bizarre ones. In fact, the more ridiculous they are, the more popular they seem to be. The Baca Robo ("stupid robot") Contest, originally started in 2007 by Maywa Denki, has now made its way to Europe by way of Budapest, Hungary. The contest has three simple tenets: people, robots, and laughter. The louder people howl at your robotic creation, the higher your score will be, bringing you that much closer to the coveted \$2,700 USD prize.



OH DEER!

Meet Robo-deer — the latest modern tool being used by Fish and Wildlife Conservation officers to nab folks who keep on killing deer after the close of hunting season. "We have a problem with people poaching," says officer Greg Stastay. "So, we're going to set up a deer for them to shoot."

That's where Robo-deer steps in. He's here to take a few bullets so his real-deer peers don't have to.

Officers are able to control Robo-deer's movements from up to 50 feet away with a radio-controlled device mounted to its back. When the tempting target is placed in the brush along the roadside, folks driving by who have no qualms about hunting illegally will inevitably stop to shoot at the "animal," — giving officers justification to step in and arrest them.

"Anybody who shoots at that deer will be arrested for illegal method and hunting deer out of season," says Stastay. Robo-deer shooters can expect a severe punishment, too — maximum penalties range between 60 days and one year in the slammer.



Photo by Stephen Messenger



ROBOT SHARE AND TELL

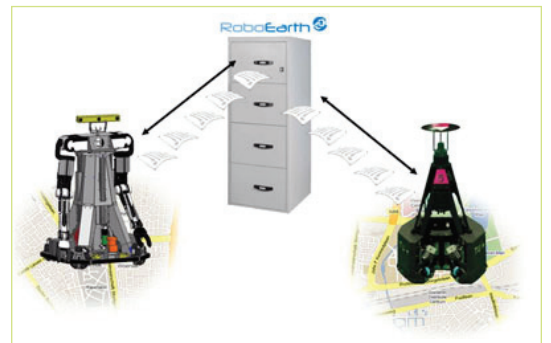
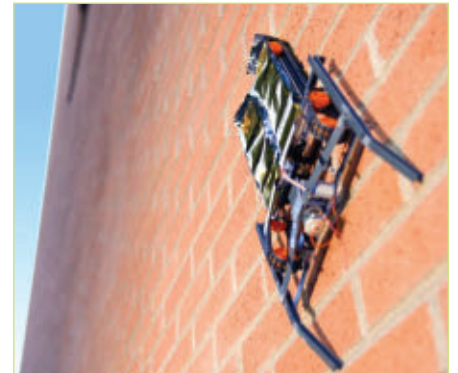
Want to share your latest robo build with others? RobotBox is a new site dedicated for this purpose. So far, the site has about 1,000 members and 200 bots. You can upload your text, images, and videos, including links to your robotic inspirations. Described by creator William Cox as an online portfolio for robot builders, members can vote on other projects and make new friends with similar interests.

AT YOUR SERVICE

KSERA (Knowledgeable SErvice Robots for Aging) is a project that concentrates on service robots for older persons who need assistance in daily activities, disease management and general care. The project — involving seven partners from five different countries — began in February 2010 and will run until 2013. There are several goals involved, one of which is RoboEarth — a network and database where robots only can share information and learn from each other. (Is that really a good idea?)

STICKY SITUATION

Based on polymer and electrostatic technologies, SRI is developing electroadhesion. It allows an object to be stuck on any surface and move around or stay put. The company says that 11 sq ft of it can support 440 lbs while using only 40 mW of power. They are hoping to incorporate the tech into robots that can travel across any terrain to inspect natural disasters, military actions, and public safety threats.





CODY BLUE

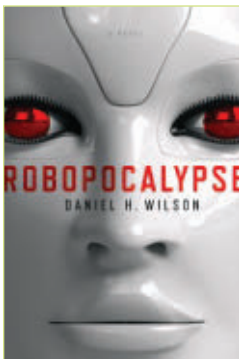
A team from the Georgia Institute of Technology has built Cody — a health service robot that can assist the elderly. He uses cameras and lasers to evaluate the patient and wipes him down with flexible arms and towels for hands. The first human to receive the care was co-creator Chih-Hung (Aaron) King, who described the experience:

"In the beginning I felt a bit tense, but never scared. As the experiment progressed, my trust in the robot grew and my tension waned. Throughout the experiment, I suffered little-to-no discomfort."



MATERIALS GIRL

Lisastarchild (yes that's her real first name, and no her parents weren't hippies!) is a handmade art toy maker from Cleveland, OH. Inspired by designer toys, urban vinyl, and the art of Luke Chueh and Buff Monster, Lisastarchild hand-sculpts figures out of polymer clay. Subscribe to her blog at <http://lisastarchild.blogspot.com>. (She gives away free stuff from time to time.)



ROBOPOCALYPSE COMING SOON

Dreamworks has announced that Steven Spielberg will begin shooting "Robopocalypse" in January 2012. Based on an as yet unpublished Daniel H. Wilson novel, it concerns human survival when the robot apocalypse begins. Wilson previously wrote *How to Survive a Robot Uprising: Tips on Defending Yourself Against the Coming Rebellion* which is also slated for Hollywood with director Steve Pink and actor Jack Black.

BOT PONG

Back in July '08, Robo-Pong cost approx. \$229 for the 540 Table Tennis Robot, but now it is a lot less. The system comes with everything you need to challenge yourself including 48 orange balls. Check it out on Amazon; sale price is \$129.



Cool tidbits herein provided by Evan Ackerman at www.botjunkie.com, www.robotsnob.com, www.plasticpals.com, and other places.

ComBots Cup V — Big Bad Bots Bash Bumpers in the Bay Area



PHOTO 1. The coveted ComBots Cup.

By Dave Calkins and Simone Davalos

Once again, combat robots from around the world made the annual pilgrimage to California to fight for the glory of the ComBots Cup — a giant trophy coveted by builders the way hockey players covet the Stanley Cup.

The fifth annual ComBots Cup combat robot championship took place October 23-24, '10 in San Mateo, CA. Dozens of old favorites, as well as some new faces, turned out for the all-combat robot event, with fighting in weight classes from tiny five ounce bots to 220 pound behemoths.

There was a prize purse of \$3,500, plus possession for one year of the ComBots Cup itself for the winner – 220 pounds of trophy-shaped stainless steel sex appeal which makes the Stanley Cup look like a Dixie cup. Roughly 130 combat robot builders from about 40 teams gathered together over the weekend in the spirit of sportsmanship, camaraderie, and all-out robotic devastation.

Builders came from as far as Brazil and the UK to see whose robot would reign supreme in each of the eight weight classes, and whose would have to go home in a small series of boxes. A total of 54 robots across the eight weight classes registered for the big event, with 13 registering for the heavyweight championship – although

only 11 of the 13 ended up being able to compete.

"We really look forward to this every year, but this year's event shows that the tournament and robots are just getting better," said Simone Davalos, co-founder of both ComBots LLC and co-founder of the International RoboGames – a springtime Olympics-style event for robots which ComBots co-organizes. "We had a great roster of robots from around the USA, and we even had a few coming from further than that. We also had really great audience turnout as well, especially considering the rainy weather."

"Combat robots as a sport is growing so dramatically, we're very pleased to see kids and adults alike get involved. For the kids, it's a great way to learn science and engineering, and for the adults it's just a lot of fun to

Class	1st	2nd	3rd
5 oz	Atom Bomb	Nano Nightmare	Rip Up
1 lb	Warpig	Darth Carbon	Sporkinok
3 lb	RamVac	Destroyer	Attitude
3 lb auton	Otto	Peanut Tin of Terror	—
30 lb	Caolho	Bird of Prey	The Bully
60 lb	The Big Bee	Texas Heat	Come to Mama
120 lb	Mortician	Wolverine 4	DoomBa
220 lb	Original Sin	Sewer Snake	Last Rites

Table 1. Winners List.

watch and participate."

Last year's winner Gary Gin with his robot "Original Sin," defended the trophy against 12 other well-known combat robot champions like Ray Billings of Team Hardcore, and Matt and Wendy Maxham of Team Plumb Crazy.

"I think the fights for the Cup were pretty darned awesome this year. We had a lot of fresh metal in the mix," observed Gin. "Before this year, there was still one thing that hadn't happened yet – there had never been a repeat Cup winner. It's good, though, because it shows that the Cup is a tough thing to win. Our aim before the event was to be the first-ever repeat Cup winner, although we had to go up against two of the three previous winners and they were both aiming for this too, so it was a very interesting tournament."

Matt and Wendy Maxham returned with Sewer Snake, winner of both the first ComBots Cup as well as this year's RoboGames gold medal. Ray Billings brought two robots for the heavyweight class – his notorious blade-spinner 'Last Rites' which won ComBots Cup II in 2008, and a new heavyweight bot "Great Pumpkin."

"We all knew it was going to be awesome. There was a pretty good field this year – more robots make everything more intense," said Billings. "Last Rites and my smaller 120 pound bot 'The Mortician' did their best to make sure everything was in serious pain – including the arena. We've upgraded the weapons systems on the bots with higher

PHOTO 2. Team Late Night Racing with #1 ranked "Original Sin."



horsepower ratings and a whole bunch more torque. Less burnout, more mashing!"

Of course, there were 10 other teams with robots who'd competed but never won the Championship, as well as a few rookies. None of them made things easy for the veteran winners. New this year was the kilt-bedecked team Scotbotics from Piedmont High in Piedmont, CA, with their heavyweight "The Ragin' Scotsman." Robots who lost to the Scotsman were given consolation buttons saying "The Ragin' Scotsman Kilt Me." The Ragin' Scotsman went two for two (two wins, two losses) which is an incredible result for a high school team competing against pros.

Returning to the heavyweight realm was long-time veteran Jim Smentowski of The RobotMarketplace webstore and creator of long-time favorite "Nightmare." Jim brought a new robot, "Breaker Box." Sadly, Breaker Box forfeited its first fight to Last Rites, but did better against both Ragin' Scotsman and Great Pumpkin, before being taken out by one of the "big three."

Also returning to the scene after a two-year hiatus was Billy Moon and Dick Stuplich, with their multi-bot "Death and Taxes." Most contestants are comfortable with a single bot, but Billy and Dick like to turn it up a notch, so two fought as one. "Death" was driven by Billy and has a nasty spinning drum and weighs in at 163 pounds, while "Taxes" was Dick's

lighter 50 pound wedge-bot. The two weighed in under 220 pounds combined, so could compete as a single entry. Those used to seeing single bot-on-bot fights were in for a treat as the two-on-one fights added a whole new

PHOTO 3. Matt and Wendy Maxham with their bot "Sewer Snake."



PHOTO 4. The Count rises out of "Pine Box."



strategy to the sport.

Long-time contestant Bradley Hanstad and Ted Shimoda of Team Think Tank also tried a two for one approach, building two full-sized heavyweights for the event. A new bot, "Double Rainbow Party" had a nasty drum and proved to be a formidable opponent. Their veteran robot VD6 was also scheduled to play, but the team decided to focus solely on their new bot, so VD6 sat in the stands and rooted for its brother bot through the two-day event.

"Vlad the Impaler II" made a return to the arena, sporting a new electric motor lifter, having dumped its pneumatic system. Sadly for Vlad's owner, the

improvements did little to improve Vlad's win record. "Evil black magic by other teams continue to foil our plans!" pined the robot's 579 year old undead owner, Vlad the Impaler I. Wearing a flowing red and black regal great coat, he and his team of minions did their best to return to the winner's podium. They might have had more luck in ancient Walachia rather than in modern California.

Team Tiki, veteran bot builders and students of Oakland's Laney College, also made their appearance in the arena with crowd-pleaser 'Pine Box.'

Seemingly an unassuming wooden oblong box when loaded in, Pine Box went to town once the matches started. Emanating smoke and theatrics, Pine Box opened up coffin-style to reveal a flame-spitting facsimile of the Sesame Street Count von Count, purple pointy ears and all. Lead builder Micah "Chewey" Liebowitz is renowned for his showmanship, and Pine Box didn't disappoint.

Two other college teams also competed and did quite well for themselves. RAM Robotics of the University of South Florida competed in the heavyweights with their undergraduate research project "Gruff." Although Gruff went one for two, it still did very well for a veteran robot. The Sierra College middleweight entry "Wolverine 4" fared much better, losing only one of its fights and finishing second over all.

Adding to the excitement was the involvement — new this year — of Royal Purple Synthetic Lubricants, a great partnership for all the chains, sprockets, and other things in a robot that need protection from friction and wear. Royal Purple is best known for their high-end motor oil which increases gas mileage and horse power in cars, but wanted to make a splash with their other lubricating products, including gearbox grease, chain lube, and other performance improving products.

"We've found that the hands-on (robotics) builder cares just as much about their projects as the hands-on automotive enthusiast," commented Royal Purple's vice-president Mark McFann. "ComBots events are the perfect venue to share our message about the advantage of upgrading lubricants."

Ray Billings added "We recently switched to Royal Purple MaxFilm for our chain and sprocket lube on our heavyweight weapons. I need things to run as smoothly as possible there,



FIGURE 5. ComBots Cup III winner "Last Rites."

obviously. I used to just use motor oil and I had to take everything apart and clean it all down after every match. It was messy and painful! Royal Purple is better than anything I have used before, and it's not slinging stuff everywhere like the oil would."

Ray's careful preparation and muss free components fell prey to that ol' black magic: physics. About halfway through the tournament, Ray's tool-steel blade sheared at the midway point in a fight against Original Sin. Suddenly weaponless, Team Hardcore Robotics was forced to regroup fast in the pits. Luckily, combat robot camaraderie came through as Matt and Wendy Maxham lent team Hardcore a new front end for Last Rites' next fight – against them!

All the teams fought hard, and both won and lost with great showmanship. Perhaps the most dramatic victory came in the 30 pound class. "Caolho," the only Brazilian entry, won all of its matches and Paulo Lenz and his comrades finally took home their first first-place win – having come to America since 2005 to compete at ComBots events. Caolho was so well built that on winning their final match, the team threw their robot across the arena in a victory cheer – where it bounced around unharmed, ready for another fight! John Frizell of the UK was the other international first-place winner, with his autonomous three pound robot "Otto."

Of course, everyone wants to know who won the big trophy and took home the check. The previous winners of the Cup continued to dominate the competition ladder, with the "big three" staying in the top three. Last Rites fell to third place, and the final round saw Gary Gin with Original Sin going head to head with Matt Maxham and Sewer Snake for the Cup. Like all the fights over the weekend, it was a great match-up, going the full three minutes. The first minute was dominated by Sewer Snake as it pushed Original Sin around the arena, but Gary's great driving came back in the final minutes, before the clock finally ran out and the match went to the judges for scoring.

While Sewer Snake had bested Original Sin for first place at this year's RoboGames, a repeat performance was not to be had, and Original Sin won by a unanimous judge's decision. Congrats to Gary Gin and all of Late Night Racing on their second ComBots Cup victory!

The ComBots Cup will return in October

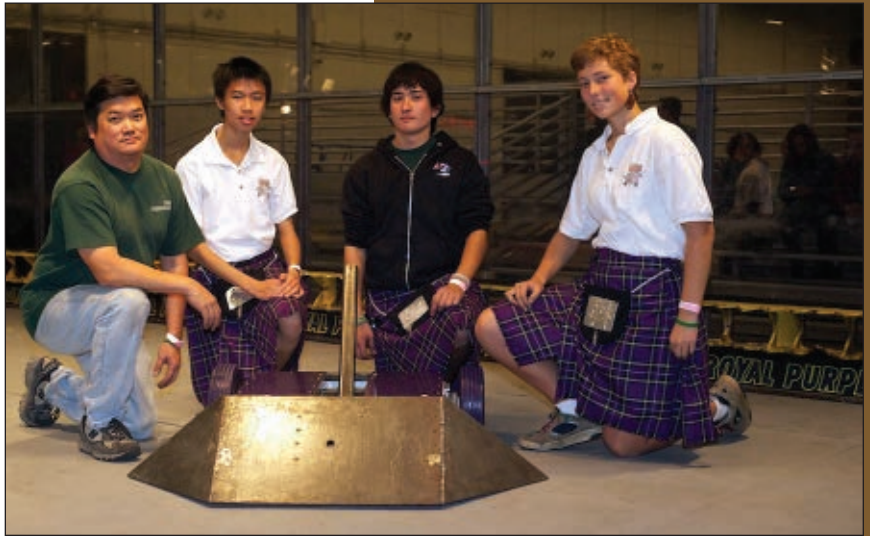


PHOTO 6. Kilt-wearing Piedmont High students and "Ragin' Scotsman."

2011 for its sixth year, with many of this year's contestants coming back for both the cold hard cash and the chance to take the trophy away from Gary and let it decorate their home for the year. With a year to design and build a robot – and save for trip – the only question to ask is will you be ready to compete and take the trophy away from Gary's grip? **SV**

www.servomagazine.com/index.php?/magazine/article/january2011_Calkins



PHOTO 7. Billy Moon and Dick Stuplich with their multibot "Death and Taxes."

par·a·digm *noun*

1: Example, pattern; especially: an outstandingly clear or typical example or archetype.

2: A philosophical and theoretical framework of a scientific school or discipline within which theories, laws, and generalizations and the experiments performed in support of them are formulated; broadly: a philosophical or theoretical framework of any kind.

merriam-webster

A New Paradigm in Hobby Robotics

by John Blankenship and Samuel Mishal

A brief look at the history of computers and microcontrollers can provide insights into the future of robotics. Based on these insights, a new and innovative approach to building and programming hobby robots is offered for consideration. Many aspects of this new proposal are already available and designs to provide additional functionality are underway. Reader's comments and suggestions for enhancing the outcome are welcome at www.RobotBASIC.com.

The methodology used by hobbyists for constructing robots has changed greatly over the last few decades. In the early years, significant skills were needed in both electronics and mechanics in order to create anything even remotely interesting, let alone exciting or useful. In the past, it was not unusual for hobbyists to build wooden platforms propelled with salvaged automobile windshield wiper motors. When motor controllers, infrared reflective sensors, or ultrasonic rangefinders were required, they had to be built from scratch and that required an understanding of technical topics such as transistor theory, phase-locked loops, passive and active filters design, and an understanding of operational amplifiers.

In more recent times, vendors such as Lynxmotion and Parallax have made available a wide variety of motors and wheel assemblies that can be mounted on pre-cut and pre-drilled aluminum or Plexiglas chassis. In addition, a wide variety of ready-to-use sensors from companies like Parallax and Pololu can now be used without a degree in

electronics, making it considerably easier to enter the field of hobby robotics.

The microcontrollers that provide the intelligence for today's hobby robots have also evolved. They come in a wide variety of sizes and abilities, and can be programmed using an assortment of assembly and high-level languages. One would think that with all these advancements, hobbyists would be dedicating more time to programming robots rather than building them. Unfortunately, there still are obstacles that must be resolved before that can happen on an appreciable scale.

Before we look at these obstacles though, let's examine why turning the focus from building robots to programming them is of such importance. We can learn a lot from the early history of personal computers. In the early 1970s, if you wanted a computer you had to build one for yourself. Hobbyists of that era wanted to learn about programming their machines, but unfortunately, with all the time required to build and debug hardware, little time was left for programming. Even when time was available, the lack of appropriate programming languages

and tools often made the endeavor prohibitively cumbersome. When companies like Apple, RadioShack, and IBM started selling fully assembled computers, hobbyists could finally turn their attention to programming. Programming tools and better languages and operating systems were some of the first advances to aid the average user. This paradigm shift facilitated the development of databases, spreadsheets, and WYSIWYG (what you see is what you get) word processors, and in the years that followed, programmers created amazing applications that revolutionized the entire computer industry.

You may wonder how all this relates to robotics. As mentioned earlier, advances in and the availability of both motor control and sensor technology have made building robots easier than ever. Unfortunately, even with these advances, there are still obstacles limiting the transition to programming.

The languages available for most microcontrollers are powerful tools in the hands of experienced programmers. Nevertheless, the physical designs associated with most microcontrollers (limited program memory and data storage, and a lack of hardware support for floating point variables, etc.) force the languages used to program these microcontrollers to be less than full-featured and often cryptic in nature. Such languages are acceptable for programming simple algorithms, but if complex behaviors and artificial intelligence are to be achieved, more capable high-level tools are needed.

Advances in and the availability of both motor control and sensor technology have made building robots easier than ever. Unfortunately, even with these advances, there are still obstacles limiting the transition to programming.

One language that makes developing and debugging behavioral algorithms easier and more efficient is RobotBASIC (RB). Beginners can use its pseudo-code like syntax to be productive in record time, and more advanced users will enjoy the C-style syntax options and the advanced functionalities that make it easy to handle complex tasks. Refer to the **sidebar** for a list of the highlights of this free language.

Since programs written in RB run on a PC (instead of on a microcontroller) where there is plenty of processing power and memory, users get the benefit of unlimited multi-dimensional arrays, floating point variables, and a plethora of unique commands and functions that handle many tasks that are difficult — if not impossible — to achieve in microcontroller languages.

When it comes to developing algorithms for robotic behaviors, perhaps one of RobotBASIC's most powerful tools is an integrated robot simulator. While the two-dimensional simulated robot may not seem realistic at first, the variety of sensors on the robot and the method with

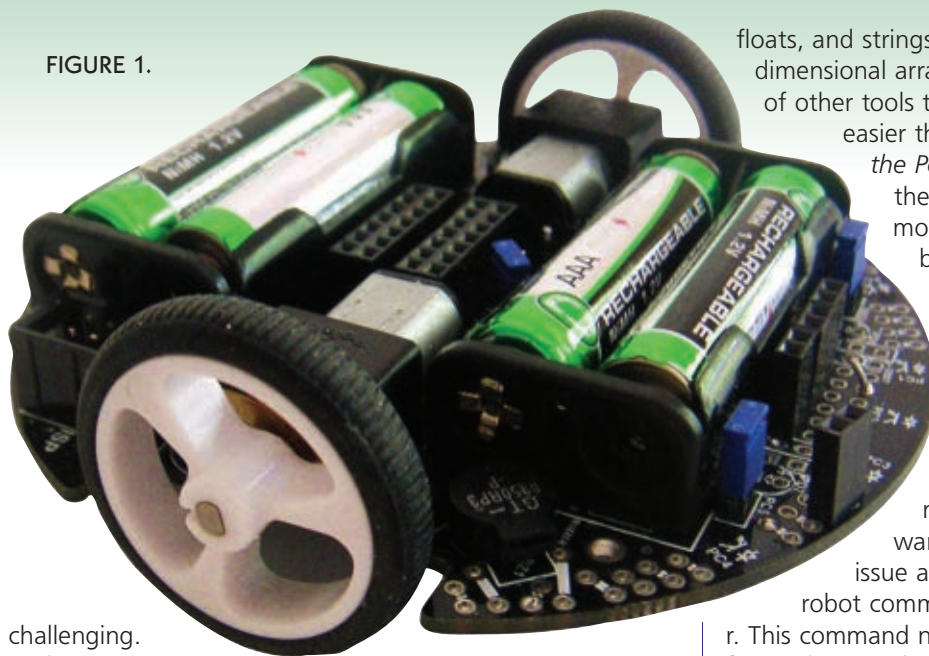
Highlights of the RobotBASIC Language

- **No installation is required** – you can run RobotBASIC from a USB drive, a CD, or even from a web page.
- Use variable typing (integer, float, and string) OR totally untyped variables where ANY variable at ANY time can be changed to ANY type by simply assigning it a new value.
- Standard GOSUB routines or callable function-like routines (Call/Sub) with LOCAL variable scoping with by reference and by value parameters (which may be skipped), as well as a returned value.
- Use legacy INPUT and PRINT statements for QUICK and EASY I/O (great for introducing programming to non-programmers).
- GUI commands that create buttons, text boxes, edit boxes, list boxes, dialog windows, message boxes, radio buttons, check boxes, sliders, and more.
- Use a Procedural programming model with GOSUB and CALL/SUB or an EVENT-DRIVEN approach with commands like OnSlider.
- Use standard BASIC syntax or a modified C-style syntax (e.g., ++, +=, !=, &&). This can be great for teaching programming fundamentals before moving on to more complex principles and syntax.
- Increased productivity from numerous “helper functions” that facilitate sorting, multimedia displays, flicker-free 2-D and 3-D animation, robot vision (including web cam support), extensive BMP image manipulation, matrix math, both high and low-level file I/O, the ability to send emails (SMTP) and communicate over the Internet (UDP and TCP protocols), and much more.
- Develop and debug programs in an easy-to-use INTERPRETER-based IDE (Integrated Development Environment), then COMPILE your programs to standalone EXEs for easy distribution.
- Ability to create includable LIBRARY files with #include (even include binary files to protect your algorithms).
- An integrated robot simulator with numerous sensors such as an electronic compass, ultrasonic distance measurement, IR perimeter sensing, line sensors, a GPS, and more.
- Control real robots using parallel, serial, and USB ports for wireless protocols such as Bluetooth and Zigbee.
- Unique proprietary protocol allows simulator programs (without modification) to control real robots (see our books and the HELP file for complete details).
- Direct support for the USBmicro U4x1 family of I/O modules that provide 1-wire, SPI, and I²C serial control of digital microdevices, as well as 16 lines of TTL I/O.
- Over 800 commands and functions often allow a few lines of code to provide the functionality of hundreds of lines in many other languages.
- An extensive HELP file provides detailed information and numerous programming examples.
- Web page tutorials, sample programs, and YouTube videos provide free help if needed.
- Integrated editor with multiple file capability and color-coded keywords.
- **RobotBASIC is FREE** to schools, organizations, and individuals — EVERYONE basically!

which they are used and programmed provide a realistic life-like behavior. The robot has bumper sensors, perimeter proximity sensors, line sensors, ranging sensors, a beacon detector, an electronic compass, a battery-level sensor, and a simple GPS system.

Debugging code on a real robot has always been

FIGURE 1.



challenging.

Faulty programs can easily damage the robot itself and the erratic operation of untested sensors can make debugging a new algorithm a nightmare. RobotBASIC's simulated robot and integrated debugging tools address both of these problems.

You may presume that RobotBASIC's simulation capability is only valuable as a prototyping tool. You can certainly develop and debug a behavioral algorithm using the simulation and then translate the principles associated with the finished program into the native language for your preferred microcontroller, but RB provides a new approach for controlling your robot — **a new paradigm for hobby robotics.**

It is worth noting that many industries have successfully used the principle of *simulate then deploy* for decades, so it makes sense that this approach be applied to robotics. Microsoft's Robotic Studio (RS) also follows this paradigm, but its complexities tend to make it more suitable for professional developers than hobbyists. One reason RS is so complex is that it supports an endless variety of configurations. RB achieves simplicity by limiting the robot's sensory system to a proven — yet predefined — set of sensor types and placements. The book *Robot Programmer's Bonanza* demonstrates sensor arrangements chosen for RB are suitable for implementing many robotic behaviors.

RobotBASIC has a built-in protocol for controlling a real robot using the **very same** programs used to control the simulated robot. The control can extend over a Bluetooth or other wireless link for small robots, or a netbook computer running RobotBASIC can be embedded in larger robots. Either method lets you program your robot using a full-featured language with virtually unlimited variable space for integers,

floats, and strings. You will have access to multi-dimensional arrays, trigonometric functions, and a host of other tools that can make programming a robot easier than ever before. A new book, *Enhancing the Pololu 3pi with RobotBASIC* shows both the hardware and software details of how most of RobotBASIC's simulated sensors can be implemented on a real robot. **Figure 1** shows a photo of the standard 3pi robot from Pololu and **Figure 2** shows a modified 3pi fully loaded with sensors that correspond to the simulated sensors available on RB's simulation.

Let's examine the basic principles used by RobotBASIC to control the modified robot. When a RB program wants to move the simulated robot, it might issue a command such as `rForward 20`. Note: All robot commands in RobotBASIC start with the letter

`r`. This command normally moves the simulated robot forward 20 pixels which is the default radius of the robot. A command such as `rTurn 10` would turn the robot 10° to the right. Functions such as `rFeel()` and `rBumper()` interrogate the simulated sensors on the robot and return values that correspond to the current environmental situation.

The above commands and functions make it easy to control the simulated robot with appropriate algorithms. To have these very same commands and algorithms control a real robot, you simply add the command `rCommPort N` at the beginning of the program. This command indicates that a Bluetooth adapter (or other serial communication device) is attached and using serial port **N** (typically a virtual port).

Once the `rCommPort` command has been issued, all RB's robot-related commands and functions (including `rForward` and `rTurn`) no longer control the simulation.

Instead, they *automatically* send two bytes to the specified serial port. The first of these bytes is an op-code that identifies the command. The table in **Figure 3** shows the op-code used for each of the simulator commands implemented on the **enhanced** 3pi robot.

The second byte sent to the robot is either zero — if not needed — or a parameter related to the command. For example, when the command `rForward 20` is issued, the PC will send out a 6 followed by a 20. The 6, of course, indicates the FORWARD operation is being requested and the 20 specifies how far forward to move. The units for the real robot obviously cannot be pixels, but we must maintain compatibility if the behavior of the real robot is to mimic the behavior of the simulation. Since a movement of 20 pixels represents the radius of the simulated robot, the code running on the real robot should move it a distance equivalent to the real robot's radius when an `rForward 20` command is received. All

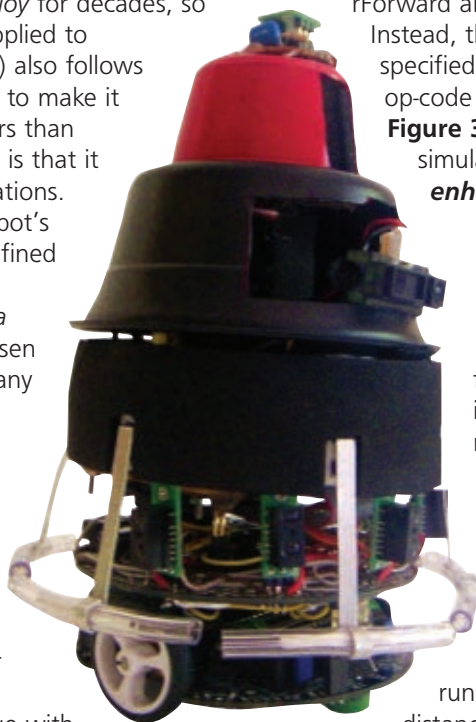


FIGURE 2.

In addition to receiving commands from the PC, the programming implemented on the microcontroller of the external robot has to return sensory data to RB. RobotBASIC's communication protocol requires that the robot return five bytes of sensory data every time it receives a

The remaining two bytes of the returned data are usually zero because they are typically not needed. When functions such as `rCompass()`, `rBeacon()`, and `rRange()` are executed though, these two bytes are used to return the requested data. Remember, the above functions normally return the state of the simulated robot's sensors. After an `rCommPort` command has been issued though, each of these functions will return the status of the real robot's sensors. Let's look at the example program in **Figure 4** to help clarify these ideas. Notice the first line of the program is commented out for now.

When the first line is un-commented allowing the `rCommPort` command to be executed, the program will automatically control the 3pi robot causing it to perform like the simulated robot — that is, it moves until it encounters an object, then turns away. As long as the robot's behavior is determined by sensory data (as it is in this example), the real robot will respond like the simulation. Examples of such sensor-controlled behaviors are following a line, hugging a wall, finding a doorway, tracking a beacon, and negotiating a maze.

```
//rCommport 40

rLocate 400,300

while true

    while rFeel() = 0

        rForward 1

    wend

    rTurn 150+random(60)

wend
```

FIGURE 4.

FIGURE 4.

Command	Op-code
RLocate	3
RForward	6
(backward)	7
RTurn(right)	12
(left)	13
rCompass	24
rBeacon	96
rRange (right)	192
(left)	193
rSpeed	36
rChargeLevel	108

FIGURE 3.

There are no files to compile and nothing to download. If several people in a robot club or school classroom have RB compatible robots, they can easily share entire programs, or even library routines of behaviors that they develop with their robot or just using the simulator alone. This sharing is possible even if each of the robots uses totally different microcontrollers.

It is worth noting that many industries have successfully used the principle of *simulate then deploy* for decades.

The embedded code for different robots might be totally different of course, but each robot will respond in the same manner to the algorithm implemented by the

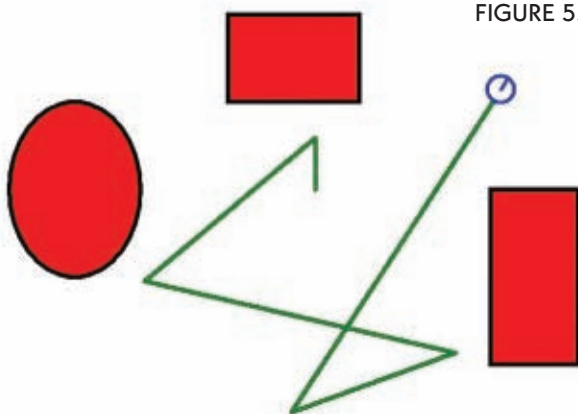


FIGURE 5.

RobotBASIC program. If you are interested in how the external robot is programmed, the source code for the 3pi robot is available for free download at www.RobotBASIC.com.

It is important to realize that the robots being controlled through this approach can be of totally different sizes and using different kinds of sensors, motors, and microcontrollers. As long as appropriate sensors are chosen and mounted like those on the simulated robot, the embedded firmware isolates application programmers from nuances that do not concern them, thus enabling them to spend more time on algorithmic development.

This approach to building and controlling a robot may not fit everyone's needs, but it provides advantages that are worth examining. The RobotBASIC team is dedicated to making it easy for hobbyists to build their own RB compatible robots and principles learned from the 3pi project are already being used to design a RobotBASIC Robot Operating System (RROS) that is expected to run on a Parallax Propeller-based robot controller board.

Once the controller board and the RROS are finalized, motors and sensors of various sizes and types from a variety of manufacturers can be connected and utilized with only a minimal knowledge of electronics. The RROS will automatically control the motors and gather sensor data, mapping it to the RobotBASIC functions so that the entire process is seamless, making programming a real-life robot no harder than programming the simulated one. **SV**

1-800-422-1100

Open the doors to a career in Electronics get YOUR keys Today!

View Free Lessons at:
www.iLearnElectronics.com
www.GSSTechEd.com

AndyMark
Inspiring Mobility

**Specializing in Unique Wheels,
Gearboxes, Aluminum Sprockets
and Drive Bases**



**6" Aluminum Dualie
Omni Wheel**



Tri- Lambda Drive Base
Omni-directional drive
system kit.



8" Mecanum Wheel



Aluminum Sprockets

AndyMark, Inc.
700 E. Firmin St., #114
Kokomo, IN 46902

765-868-4779

www.andymark.com

AP CIRCUITS
PCB Fabrication Since 1984

As low as...

\$9.95

each!

Two Boards
Two Layers
Two Masks
One Legend

Unmasked boards ship next day!

www.apcircuits.com

VISA MasterCard PayPal IPC MEMBER BBB



By Fred Eady

Experience the Emperor Pinguino

If the word “Arduino” is foreign to you, odds are that you’ve been exiled to a hole in the ground in the unexplored depths of the Amazon jungle. I recently came face to face with an Arduino while helping some senior aeronautical engineering students complete their senior design project. Not one of the aeronautical-engineers-to-be considered themselves as “programmers.” However, they all adapted to the Arduino environment quite handily.

The Arduino platform the students used intrigued me as I witnessed a bunch of college kids take an unknown entity, learn to use it, and apply it to their control application. I decided to take a serious look into the “Arduino effect.” Arduino is touted as an open source platform and I was pleased to find a couple of Arduino variants that were out of the ordinary. The Arduino variant that caught my eye is really not an Arduino platform at all. It is called Pinguino.

Pinguino 101

I’m not going to get up on my donkey and make the assumption that all of you reading this are intimately familiar with the Arduino environment and the physics that make it work. In that Pinguino is Arduino-compatible, we’re going to concentrate on getting Arduino-like results from Pinguino hardware and software. Currently, Pinguino hardware is based on the PIC18F2550 and PIC18F4550. Pinguino software is very similar to the Arduino and is built from open source packages.

Although the results of AVR-based Arduino designs are for the most part identical to the results one would expect to receive from a Pinguino design, the Pinguino hardware is very different in content. A typical piece of AVR-based Arduino hardware is based on an Atmel ATmega8, ATmega168, or ATmega328. Larger AVR microcontrollers are also being integrated into the Arduino environment. As of this writing, the flagship Arduino board is the Uno which is based on the ATmega328. The new Arduino Uno differs from its ancestors in that the FTDI USB-to-UART IC has been replaced with an ATmega8U2 USB-enabled microcontroller.

The advantages of using an AVR instead of the FTDI part are speed and USB class flexibility. The new Arduino platforms that are front-ended with a USB-equipped microcontroller can be configured as HID-class USB devices, as well as CDC-class devices.

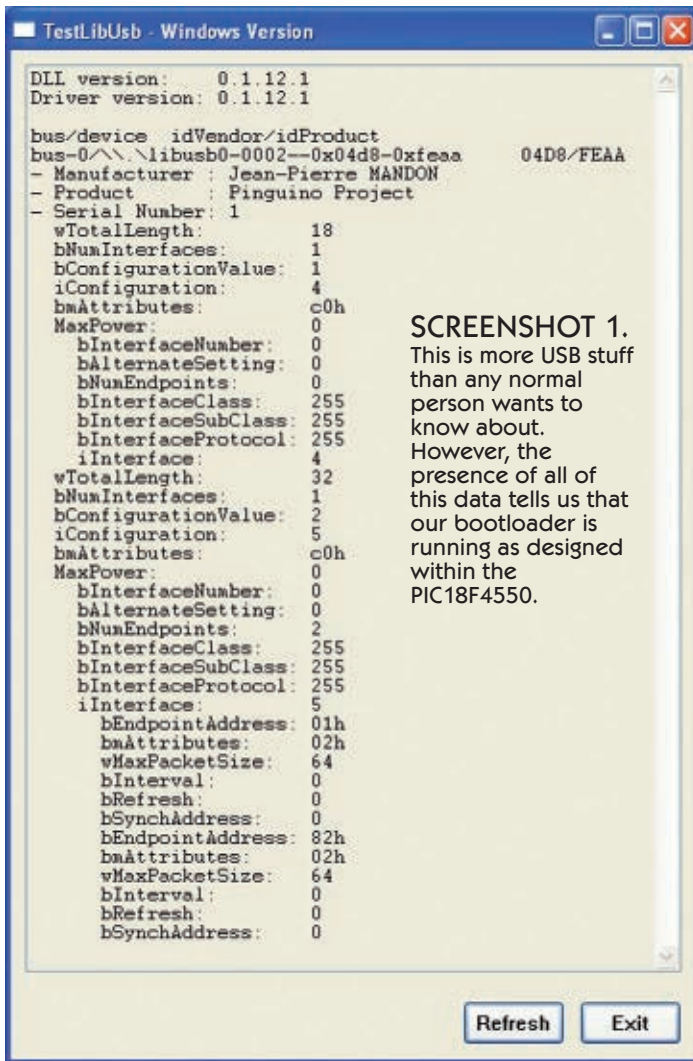
You can beat the AVR and PIC datasheets against each other to pull out the fine points of differences in the microcontrollers. However, in a nutshell, the Pinguino platform is based on a USB-enabled PIC and does not require an additional USB front-end device. The PIC18F2550 is a 28-pin USB-enabled PIC and offers up to 18 I/O pins that fall under the control of the Pinguino environment. The PIC18F4550 is available in 40-pin DIP and 44-pin TQFP packages and — if you include the RUN LED I/O pin — brings 30 I/O pins to the Pinguino party.

The Pinguino environment is designed to operate with the same C-type mnemonics used in the Arduino programming environment. That makes sense as the Arduino and Pinguino are both based on open source code. Both the Arduino and Pinguino platforms leverage open source to allow them to run on all of the popular operating systems which include Linux, MAC OSX, and Windows XP/7.

As I alluded to earlier, the Pinguino programming language is very similar to C. However, there is no *C main* function. The *C main* function is replaced by a *loop* function. A *setup* function is also part of the Pinguino programming model. The *setup* function runs only once following a hardware or power-on reset. Here’s a super simple example of how the *setup* and *loop* functions are used to blink an LED attached to one of the Pinguino microcontroller’s I/O pins:

```
#define PIC18F4550

void setup()
{
```

SCREENSHOT 1.

This is more USB stuff than any normal person wants to know about. However, the presence of all of this data tells us that our bootloader is running as designed within the PIC18F4550.

```
pinMode(21,OUTPUT);
}

void loop()
{
  digitalWrite(21,HIGH);
  delay(500);
  digitalWrite(21,LOW);
  delay(500);
}
```

Pin 21 is actually a Pinguino I/O pin name that associates with a physical I/O pin on the PIC18F4550. You'll see how the Pinguino I/O pin naming convention works as we continue our discussion. For now, trust me. Pin 21 is actually pin RD0 on the PIC18F4550 and it does indeed blink the LED under the control of the code we just examined. As the PIC18F4550 is not the base Pinguino microcontroller, the PIC18F4550 *#define* statement is mandatory.

Before we can blink LEDs with our Pinguino hardware, we must download and install the Pinguino software development environment. For Windows, that means downloading a copy of Python 2.5.2, wxPhthon 2.8, PyUSB 1.5, and libusb-win32. Once all of the Python and libusb components are installed, a modified Microchip USB driver must be downloaded and invoked.

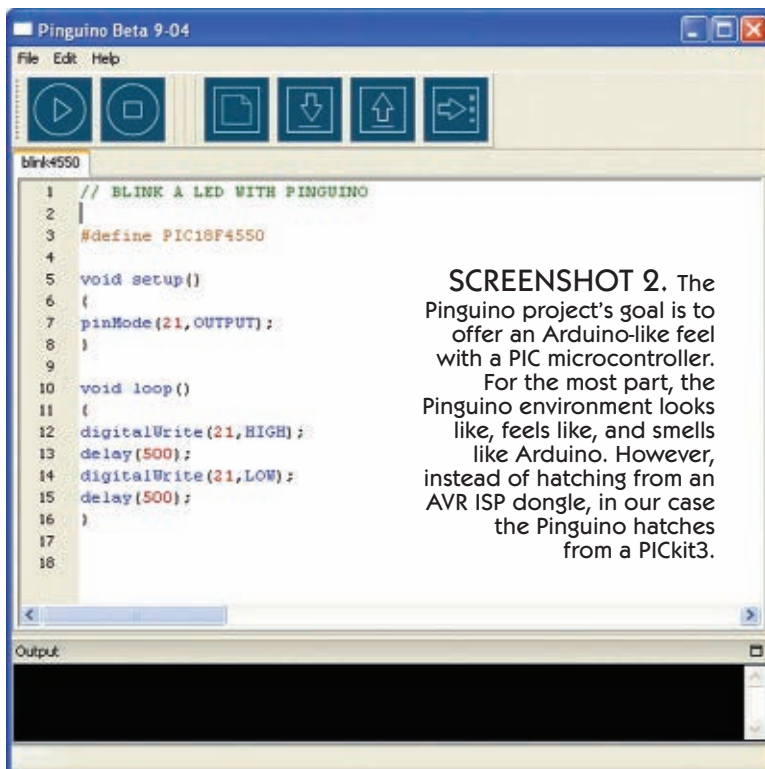
Absolutely nothing Pinguino is going to happen before we load the PIC18F4550 with the Pinguino bootloader which can also be had for a free download. If you're building your Pinguino hardware from scratch — and we are —, you'll need some type of PIC programmer to initially load the Pinguino bootloader into the PIC18F4550. I used a PICkit3 and MPLAB to import the file *bootloaderV2.12.hex* and program it into the PIC18F4550. Once the PIC18F4550 bootloader was programmed successfully, Windows

detected the new device and installed the modified Pinguino USB driver.

Following the successful installation of the Pinguino USB driver, I ran a native *libusb-win32* component called *testlibusb-win.exe* which contacted the newborn Pinguino hardware and produced the results shown in **Screenshot 1**. From now on, that newborn piece of Pinguino hardware shall be called Emperor.

Installing the latest and greatest Pinguino IDE code is all that stands between us and some serious Pinguino coding. At the time of this writing, the latest Pinguino IDE package was 9.04. If you are already Arduino friendly, **Screenshot 2** is no stranger to you. At this point, I connected the open end of R6 to five volts (5V0 in **Schematic 1**) and the open end (cathode) of USER LED D7 to the Emperor's pin 21. The **Screenshot 2** blinker code compiled without error and uploaded to our new Emperor without incident. After the built-in five second bootloader delay — which was put in place to eliminate the need for a reset button — the Emperor's RUN LED illuminated and the Emperor's USER LED began to blink. That was too easy!

In the meantime, I'm powering our new



SCREENSHOT 2.

The Pinguino project's goal is to offer an Arduino-like feel with a PIC microcontroller. For the most part, the Pinguino environment looks like, feels like, and smells like Arduino. However, instead of hatching from an AVR ISP dongle, in our case the Pinguino hatches from a PICkit3.

Emperor hardware via its USB portal. Another look at **Schematic 1** indicates that the Emperor's USB PWR LED (D5) should be shining bright and it is. With no voltage emanating from the output of the LM340S-5.0, the DMP2123L-7 P-channel MOSFET is biased *ON* via resistor R2, and USB power is directed through diode D3.

Applying the appropriate voltage to the LM340S-5.0 input will result in 5.0 volts appearing at the gate of the DMP2123L-7 which turns the MOSFET *OFF*. The supply voltage (5V0) is now being sourced via the LM340S-5.0 and Schottky diode D2. As you can see in **Schematic 1**, status LEDs illuminate to signal which power source is active. Also note that the Emperor's PIC18F4550 is labeled in Pinguino speak. For instance, instead of calling pin 37 of the Emperor's PIC18F4550 RD0, it is Pinguino I/O pin 21.

This gets better. Consider this Pinguino program:

```
#define PIC18F4550

#define RUNLED PORTAbits.RA4

void setup()
{
}

void loop()
{
  RUNLED = 1;
  delay(500);
  RUNLED = 0;
  delay(500);
}
```

Following a successful compile and upload, the Emperor's RUN LED is now blinking. The *PORTAbits.RA4* reference follows Microchip's C18 C compiler syntax to the tee. Hmmmm ... Now that you've seen a bit of what the Emperor can do, let's get about building one for you.

Scratch Building an EDTP Emperor

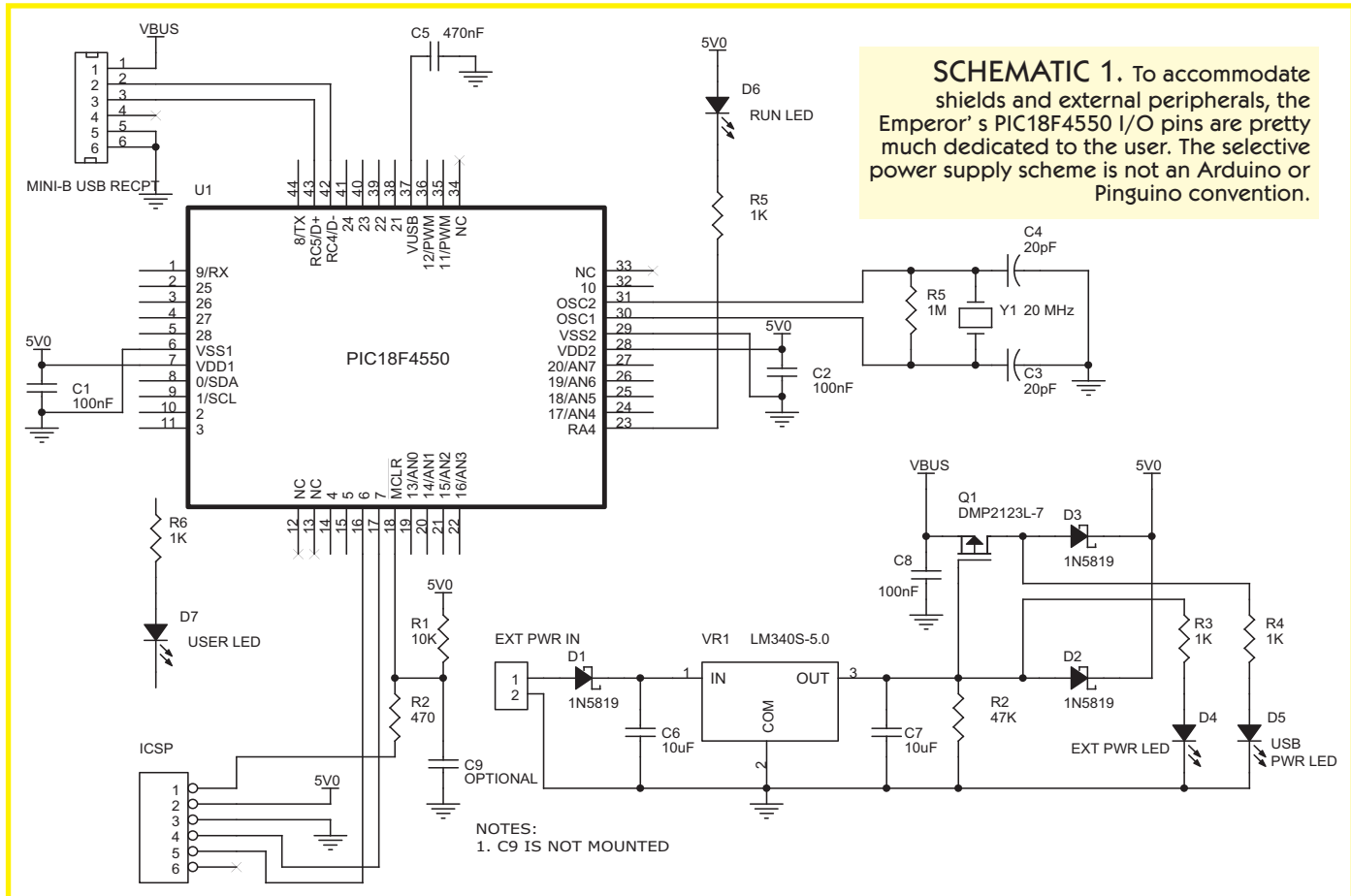
The hardware that supports the Pinguino programming environment is just as easy to replicate as the Pinguino's C-like-Arduino-like programming language. If you visit the Pinguino website, you'll find that a base PIC18F2550 Pinguino is packaged in a standardized form factor. The PIC18F4550 version of the original Pinguino design is also assembled in a DIP-like format. In that the Pinguino project is open source, we're going to blaze our own form factor path. Our Pinguino-compatible Emperor hardware is sliding

Fred Eady can be reached via email at fred@edtp.com.

Sources

Emperor
EDTP Electronics, Inc.
www.edtp.com

Pinguino Information
HackingLab
www.hackinglab.org



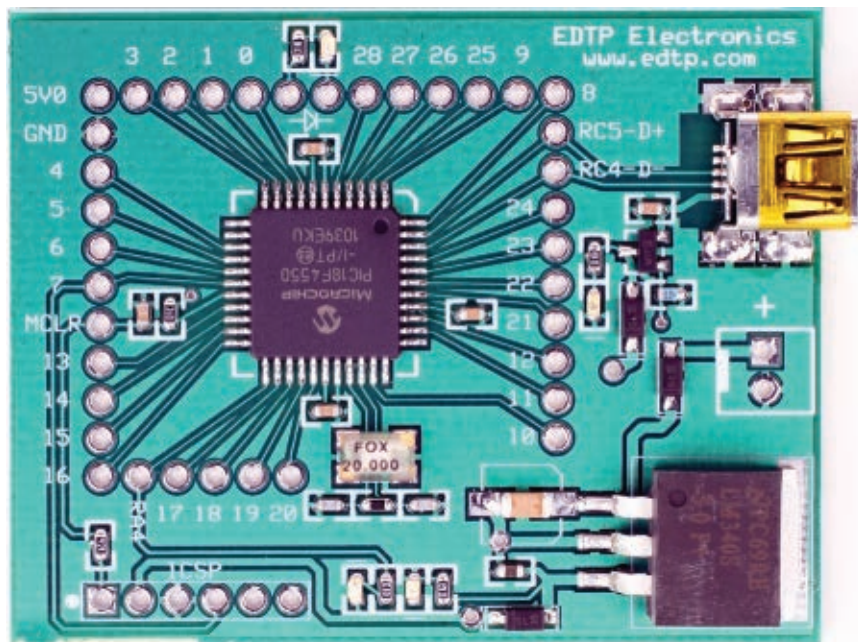


PHOTO 1. Our EDTP Emperor hardware includes special power supply circuitry that helps keep the magic smoke inside of the PIC18F4550 and your PC's USB portal.

voltage source. During testing, I was able to continually switch between the USB and external power sources without the switchover resetting the PIC18F4550.

The Emperor ICSP Circuitry

The Emperor's ICSP pads need not be populated with a header or socket. The six-hole area is designed to temporarily accommodate the short end of six standard 0.1 inch male header pins that are stuffed into the business end of a PICKit3. Note that I said "temporarily" accommodate. With the PICKit3-mounted header pins plugged into

on the ice in **Photo 1**.

The core of our Pinguino-compatible Emperor hardware is the PIC18F4550. Unlike its Arduino cousins, our Emperor hardware design consists of only a single USB-enabled microcontroller. As I stated earlier, the latest version of the official AVR Arduino hardware platform includes a front-end USB-enabled microcontroller that supplies the USB connectivity for the primary host microcontroller. You will need a PIC programmer capable of supporting the PIC18F4550 to initially load the bootloader code. By the way, if you were building an Arduino from scratch with a virgin AVR, you would also need an AVR programmer to initially load the bootloader.

The Emperor Power Supply Design

It is a big NO NO to source voltage to the host USB portal. So, our Emperor power supply design uses Schottky diodes and a P-channel MOSFET to safely route the 5.0 volt power to the PIC18F4550 and any externally attached peripherals. External peripherals under the control of the Emperor running Pinguino code are called shields in the Arduino world.

Schottky diode D1 protects the input of the LM340S-5.0 against reverse polarity power that may accidentally be applied to the external power source pins. You will need to provide a minimum of +7.0 volts at the LM340S-5.0 input pin to obtain a stable 5.0 volts at the LM340S-5.0 output pin. The voltage regulator input capacitor (C6) is a 10 μ F ceramic type housed in a 1206 package and is rated at 25 working volts. The 10 μ F output capacitor is also a ceramic device but is packaged as 0603 and rated at 10 working volts.

We've already discussed the theory behind the P-channel MOSFET and its associated Schottky diodes. As you can see in **Schematic 1**, the DMP2123L-7 allows the external input voltage source to override the USB portal

the ICSP holes, the PICKit3 can make electrical contact with the ICSP hole area only when programming or debugging is required.

Being able to temporarily attach the PICKit3 allows you to use the pair of PIC18F4550 programming pins in your application without having to isolate them from the programmer/debugger interface. For instance, if you populate the Emperor I/O pin headers to allow the Emperor to be removed from the shield it is driving, you can program the Emperor while it is stand-alone, and resocket the programmed Emperor into your shield and utilize the PIC18F4550's Pinguino-assigned ICSP pins 6 and 7 in your application.

The PIC18F4550's MCLR circuit is standard PIC fare. The reset capacitor C9 is optional. If you have reset problems, you can mount C9. A standard value for C9 is 100 nF. However, if you look at the original Pinguino schematic, you'll see C9 as a 10 μ F aluminum electrolytic. A 10 μ F 0603 ceramic capacitor can be placed in the C9 position if your environment requires it.

The Emperor's PIC18F4550

I'm not going to insult your intelligence and recite from the PIC18F4550 datasheet. Most likely, the PIC18F4550 was chosen for Pinguino service because of its on-chip USB capabilities. If you stand an ATmega328 next to a PIC18F4550, you'll also find that they are similar with respect to raw resources. Both the AVR and PIC contain 32K of self-programming Flash. If the bootloader is utilized as originally intended, self-programming Flash is essential to both the Arduino and Pinguino platforms. The opposing parts are also equals when it comes to volatile memory with both microcontrollers supporting 2K of SRAM. The AVR stands out with 1K of EEPROM versus the 256 bytes offered by the PIC18F4550. If analog-to-digital capability is high on your list, you'll choose the PIC18F4550 as it offers

an analog-to-digital system with 13 channels of 10-bit resolution versus the AVR's eight channels of 10-bit resolution with a TQFP package and six channels in the PDIP package.

In the timer department, it's almost a wash with the AVR bringing a pair of eight-bit counters and a 16-bit counter to the table. The PIC18F4550 counts with a single eight-bit counter and three 16-bit timers with some of the 16-bit timers able to operate as eight-bit timers. Both the AVR and PIC18F4550 house at least one analog comparator, an SPI portal, a USART, an I²C interface, and some PWM channels. The PIC18F4550 barely outdistances the AVR in the available I/O pin department and both microcontrollers can run with a 20 MHz clock. With that, you can easily see why the PIC18F4550 is to the Pinguino environment as the ATmega328 is to Arduino.

A Renegade Emperor Footprint

Our Emperor hardware is small enough to be placed inside of a standard Arduino shield footprint. However, our Emperor hardware is designed to mount on a standard 0.1 inch pitch perf board. Any extraordinary Emperor shield pinouts that must be accommodated can be realized with header pins or header sockets on the perf board. **Photo 2** is an example of our Emperor card mounted on an EDTP plated-through perf board with a 16 x 2 LCD. In that the Emperor footprint is small, there's nothing to stop you from laying down a shield in printed circuit board form.

We Don't Need No Stinkin' Shield

Want to drive a standard hobby servo with the Emperor? Want to do it without having to design and build a special servo shield? Well, it took me longer to hook up the servo to the Emperor than it did to write the code to drive it:

```
// HOBBY SERVO CONNECTIONS
// +---+
// |           |----- Emperor I/O Pin 17 (WHITE)
// |           |----- +5V (RED)
// |           |----- GND (BLACK)
// |           |
// +---+

#define PIC18F4550

uchar position = 1;

void setup(void)
{
    servo.attach(17);
}

void loop(void)
{
    servo.write(17,position);
    delay(500);
    position++;
}
```

Again, I won't insult your intelligence. I think you can see what and why about what it took to spin that little servo's shaft. If you've ever written any code from scratch

to drive a hobby servo, you can appreciate the Pinguino language's blunt approach. I read almost as much as I write. I know firsthand that sometimes you can't believe everything a guy like me tells you. If you have a problem believing how easy driving that servo was with an Emperor and that little bit of Pinguino speak, put an Emperor together and try it for yourself.

Big Man on the Iceberg

There are a couple of ways to slip and slide on the Pinguino iceberg. If you have a way to program the Pinguino bootloader into a PIC18F4550, you can build up an Emperor using breadboard techniques and through-hole components using **Schematic 1** as a guide. Method #2 is available to those of you that want to dedicate more time to your robot's mechanics than to a soldering iron. I will make a completely assembled and tested Emperor available to you via the EDTP website. For those of you that are PIC programming tool challenged, your EDTP-assembled Emperor will also come loaded with the Pinguino bootloader.

I've got lots of goodies on the bench that want to hop onto an Emperor shield. As far as the Emperor is concerned, this discussion is only the tip of the iceberg.

SV

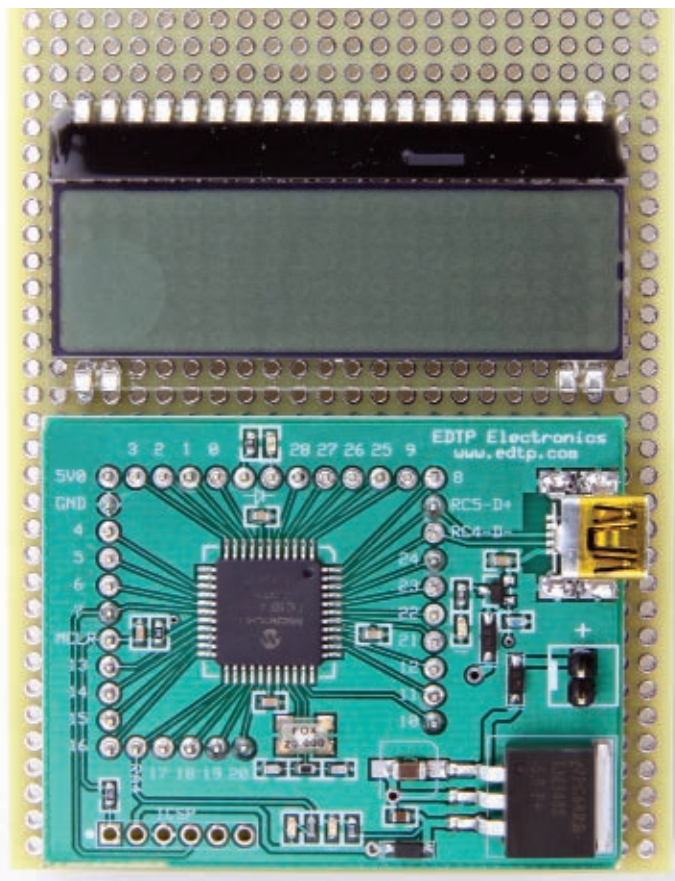


PHOTO 2. The EDTP Emperor can easily place characters on the LCD that shares this perf board. In fact, that goes for just about any peripheral device we can squeeze into the perf board's holes.

The NXT Big Thing #6

Eddie 2.0

By Greg Intermaggio



Greetings, roboteers! In this issue, we're ushering in the new year with a brand new robot design! So, bust out your NXTs and get ready to build! Eddie 2.0 is the next generation of LEGO MINDSTORMS design.

Here are just a few of Eddie 2.0's new features:

- Eddie 2.0 can be built out of just one LEGO MINDSTORMS NXT educational kit. (Eddie 1.0 required extra pieces).
- Motors have been inverted so telling Eddie to move

forward in programming no longer means he'll end up moving backwards.

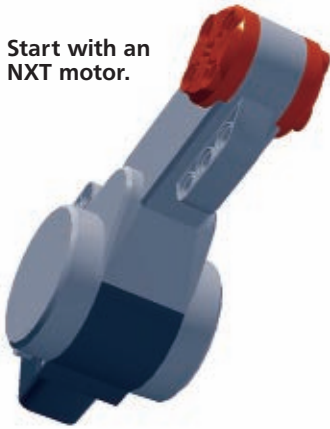
- The new design is smaller, easier to build, and has a lower center of gravity.

Let's get started!

Building Eddie 2.0

Orienting Eddie 2.0 with the ball at the back, attach the right motor to port C and the left motor to port B. Now, you're ready to rock! Try some basic programs with Eddie 2.0, re-write Eddie 1.0's programs to work for the new one, or create your own programming challenges!

- 1.** Start with an NXT motor.



- 2.** Snap in two double friction pins and slide in a four-stud axle.



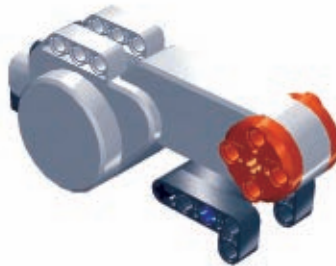
- 3.** Attach a 4 x 2 angled studless beam.



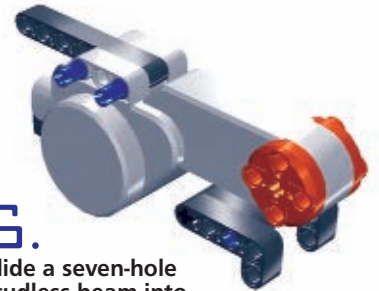
- 4.** Add a three-hole studless beam.



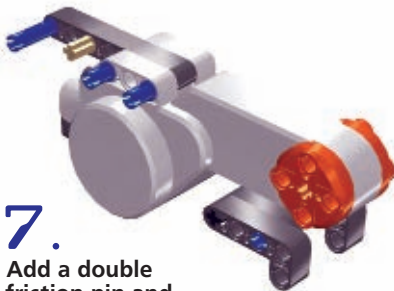
- 5.** Re-orient the motor assembly and snap on another 4 x 2.



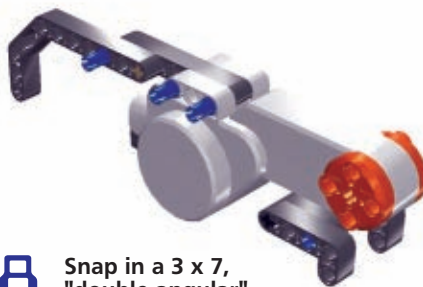
- 6.** Slide a seven-hole studless beam into place as indicated, and snap it in with two double friction pins.



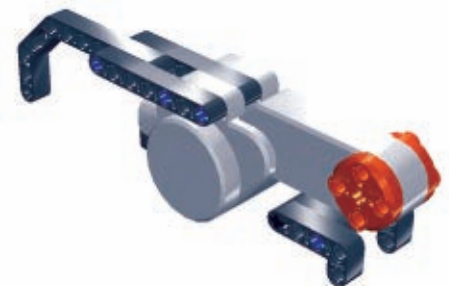
- 7.** Add a double friction pin and an axle pin.



- 8.** Snap in a 3 x 7, "double angular" studless beam.



- 9.** Attach a seven-hole studless beam on the end.



10.

Build the same motor assembly in Step 1 in reverse.



11.

Build the same motor assembly in Step 2 in reverse.



12.

Build the same motor assembly in Step 3 in reverse.



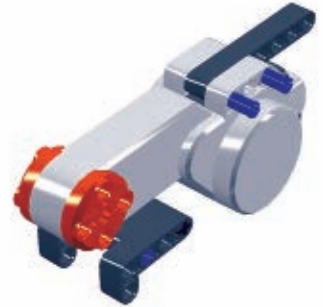
13.

Build the same motor assembly in Step 4 in reverse.



14.

Build the same motor assembly in Step 5 in reverse.

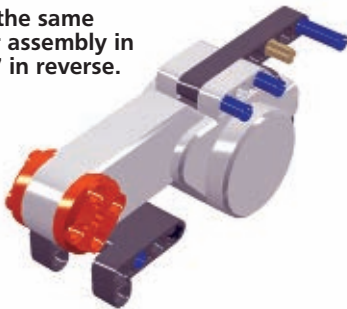


15.

Build the same motor assembly in Step 6 in reverse.

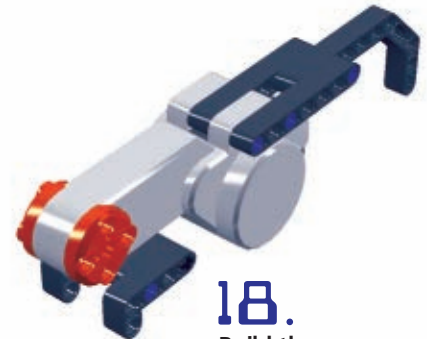
16.

Build the same motor assembly in Step 7 in reverse.



17.

Build the same motor assembly in Step 8 in reverse.

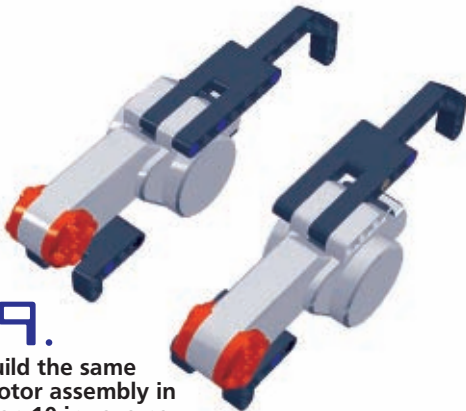


18.

Build the same motor assembly in Step 9 in reverse.

19.

Build the same motor assembly in Step 10 in reverse.



20.

Stick two double friction pins into a 3 x 5 angled studless beam.



21.

Attach long cross blocks on either side.



22.

Slip in 2x two-stud axles and 2x double friction pins as indicated.





- 23.** Line up the motor assemblies with the central assembly, and find the connector indicated.

24. Snap it all together!



- 25.** Re-orient the robot.



- 26.** Add two axle pins (make sure to use gray or tan colored axle pins which have little friction).



- 27.** Attach an axle extender to both pins.

- 28.** Slide a five-stud axle into each axle extender and a four-stud axle into each motor.

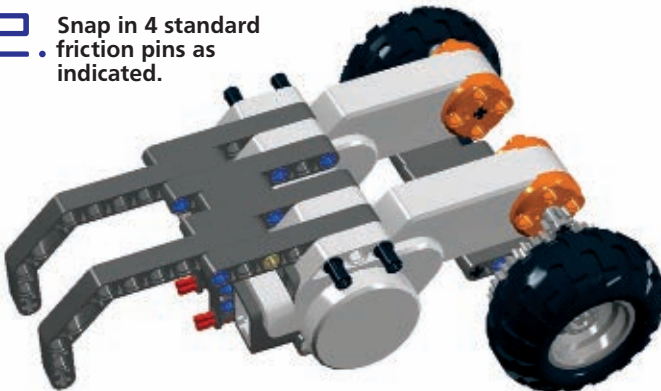


- 29.** Attach the indicated gears and half-bushings.

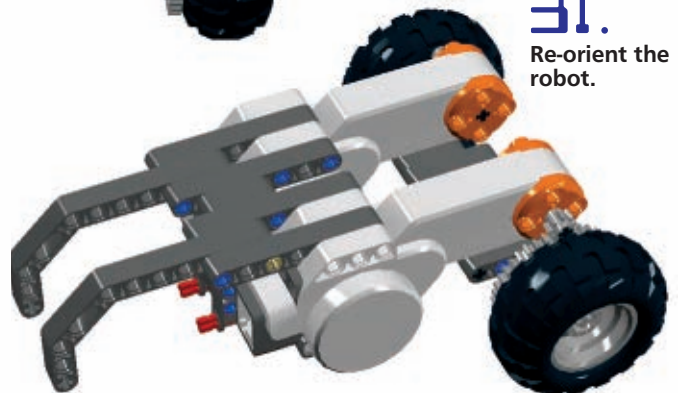


- 30.** Slip on the wheels!

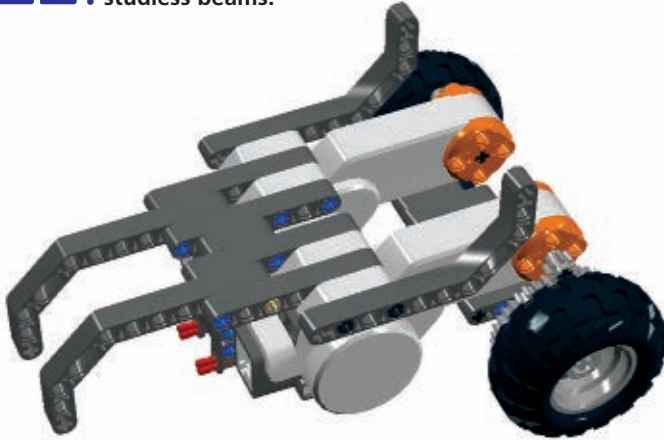
- 32.** Snap in 4 standard friction pins as indicated.



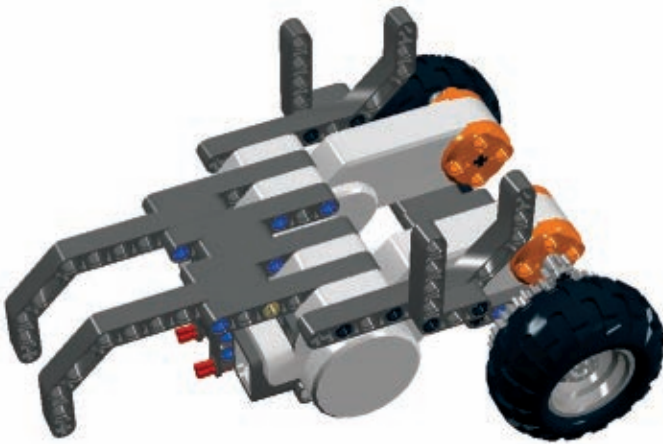
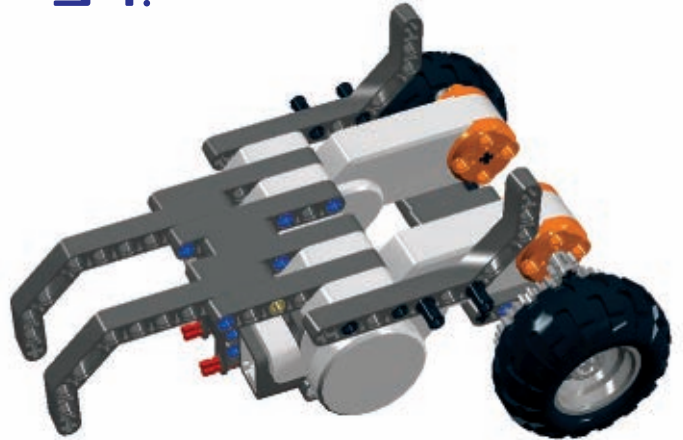
- 31.** Re-orient the robot.



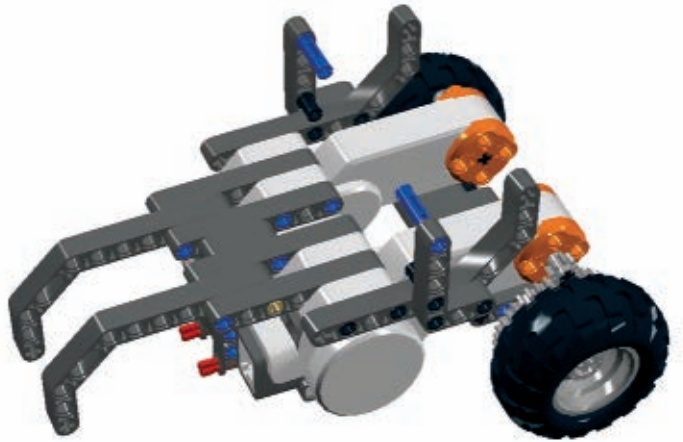
33. Snap on two 3 x 7 "double angular" studless beams.



34. Attach four more standard friction pins.

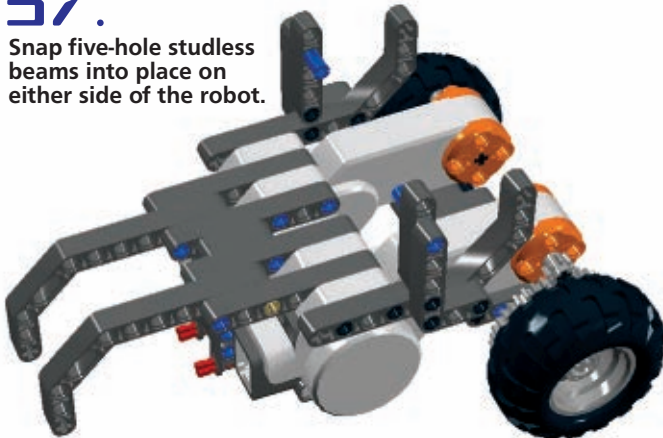


35. Slide in a 3 x 5 angled studless beam on either side.

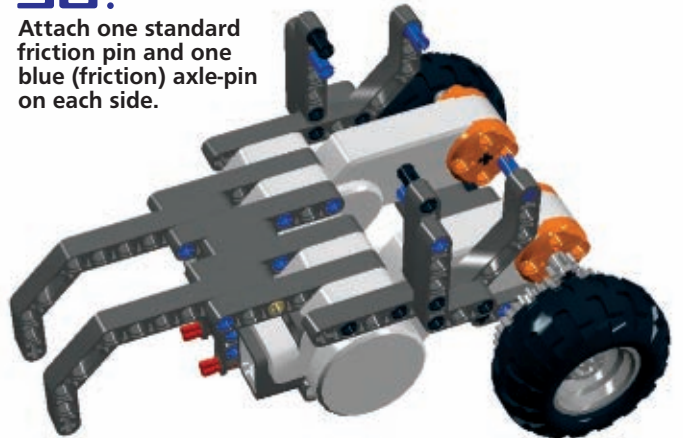


36. Attach one standard friction pin and one double friction pin on each side as indicated.

37.
Snap five-hole studless beams into place on either side of the robot.

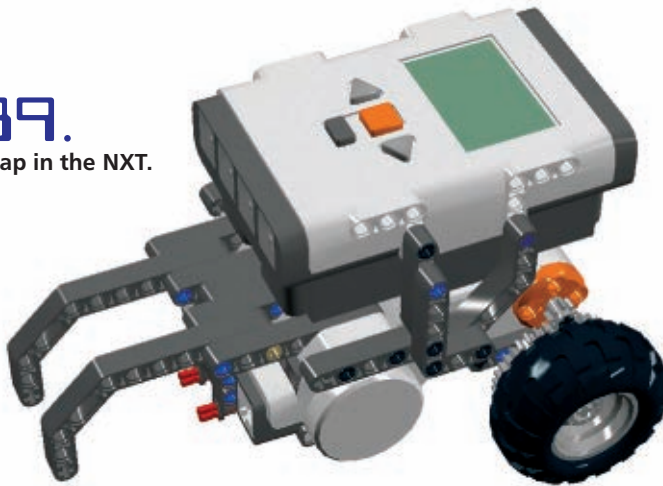


38.
Attach one standard friction pin and one blue (friction) axle-pin on each side.



39.

Snap in the NXT.



SERVO'S LEGO Demo at ComBots

October 23-24, 2010 was this year's ComBots Cup — a “mini-RoboGames” that cuts out the fluff, and gets down and dirty with full-on robot combat. ComBots Cup, as the name implies, is a combat only event, and drew roboticists from all over the world, competing in 10 different weight classes.

While heavy metal R/C robots battled it out just 10 feet away, an even more fierce competition raged at *SERVO Magazine's* inaugural LEGO robotics demo!

Kids aged 6-600 were invited to participate in a free 30-minute

workshop where they would get a chance to customize and compete with a LEGO sumo robot, just to get a taste of the whimsical world of LEGO robotics. Our own Greg “LEGO” Intermaggio fearlessly ran demo after demo with no breaks for hordes of hyperactive kids throughout the weekend.

When all was said and done, over 75 kids had participated and enjoyed our workshop, leaving not a single open slot throughout the entire weekend — a resounding success!

If you live in the Bay Area and would like to learn about Greg's hands-on science camps in your area, please send him an email at G.Intermaggio@gmail.com.

Correction From The NXT Big Thing #4:

In the fourth edition of The NXT Big Thing, we made line following robots using multiple sensors, and spent some time learning about the different types of variables. Reader Brian Burdette has pointed out a critical discrepancy in the list of variable types I provided. Here it is, with the discrepancy:

- Binary (Logic)
- Boolean (Number)
- String (Text)

Those of you who are familiar with variables and programming terminology may see the mistakes already. For those who are reading to learn, here is the corrected list:

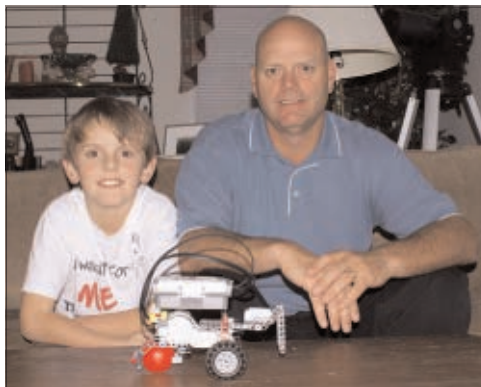
- Boolean (Logic)
- Integer (Number)
- String (Text)

Boolean variables are true/false; integer variables are numeric (1, 2, 56236, -11, etc.);

and string variables are words.

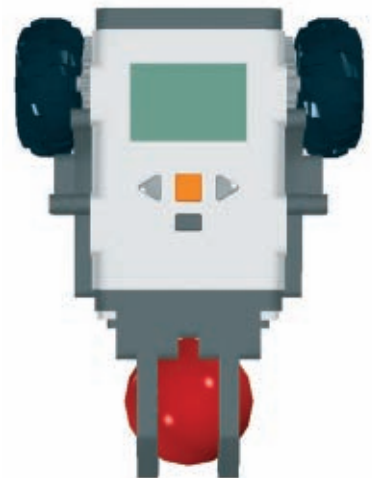
I sincerely apologize for any confusion that my mistake may have caused, and hope that this clears up any confusion on the subject.

— Greg



Brian and Carson (9 years old) Burdette
Apex, NC

40. Bird's eye view.



41. Attach a ball, and Eddie 2.0 is complete!

Wrapping Up

Next month, we'll tackle one of the greatest competitions ever devised for LEGO Robotics: Sumo. Stay tuned! **SV**

www.servomagazine.com/index.php?/magazine/article/january2011_Intermaggio

Making Robots With The

Part 3 – Inside the Arduino

By Gordon McComb

Arduino

You can construct a fully autonomous programmable robot for less than the cost of dinner and a movie for two. Mind you, I'm not suggesting one over the other — just pointing out that robots don't have to be expensive or difficult to build. It might have been true in the past, but it's not now.

That's the idea behind the ArdBot shown in **Figure 1**. It's a low cost, expandable, and easy to build mobile robot based on the popular Arduino microcontroller. Total cost of construction is under \$85, and even less if you already have some of the common components, like RC servo motors modified for continuous rotation and a solderless breadboard.

The past two installments of this series introduced the Arduino controller and the ArdBot chassis. Part 1 covered the Arduino and why this \$30 board is fast becoming a favorite among experimenters the world over. Part 2 detailed the mechanical construction of the ArdBot — a seven inch diameter desktop rover powered by replaceable or rechargeable batteries and twin RC servo motors.

This time, you'll learn more about the Arduino and its programming. The Arduino leverages a number of well supported open source projects, and mashes them into a convenient integrated development environment (IDE) that's simple to install and easy to use. In future articles, you'll apply what you learn here to the ArdBot, including

writing your own motor control functions, responding to sensor feedback, and more.

A Closer Look at the Arduino

Arduino is more a concept than it is a specific product. Since its introduction in 2005, the Arduino microcontroller board has gone through many permutations, and even today there are over half a dozen "official" Arduino boards that vary in size, shape, and capabilities — add to this literally dozens of clone Arduinos that go by other names like Freeduino, Boarduino, and many others.

Figure 2 shows the Uno — one of several Arduino boards — but one that encapsulates the core set of Arduino functionality. It's the latest version of the most popular Arduino design which features a low cost Atmel ATmega328 microcontroller mounted on a handy "stackable" development board. There are other

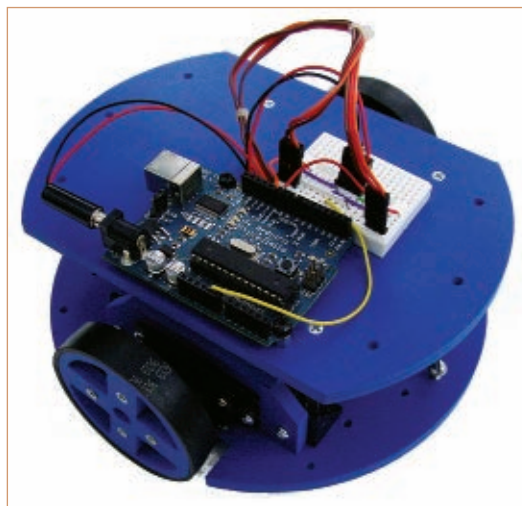


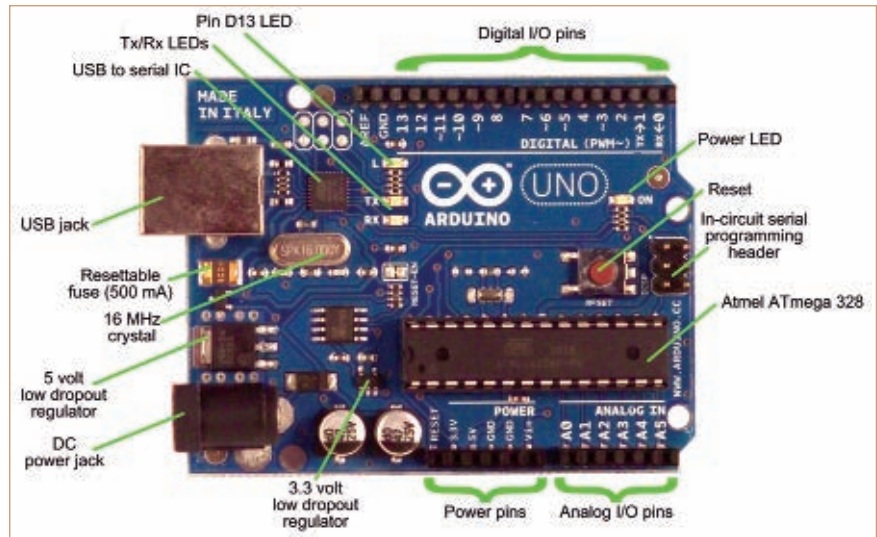
FIGURE 1. The ArdBot, with Arduino microcontroller and mini solderless breadboard for experimenting.

FIGURE 2. Pictorial overview of the main points of interest on the Arduino Uno microcontroller board.

versions of the Arduino — bigger and smaller — but it's the 2-1/8" by 2-3/4" Uno that most people use, and the one selected for the ArdBot. (If you already have an earlier version Diecimila or Duemilanove, then those are okay, too, as long as you use version 0017 or later of the Arduino IDE.)

Main points of interest of the Arduino Uno include:

- **The ATmega328 microcontroller, running at 16 MHz.** The board uses the DIP version of the ATmega328 so that if you "let the smoke out" of the thing, you can easily replace just the chip, rather than buy a whole new Arduino board.
- **Reset pushbutton.** Press to reset the currently running program.
- **Integrated USB-to-serial communications, for both downloading programs from your PC and for serial communications back to the PC for debugging and monitoring.** The USB link includes a 500 mA resettable fuse to guard against possible damage caused by a wayward Arduino to the USB ports on your PC. When plugged into a USB port, the Arduino takes its power from it. With USB 2.0, drive current is limited to 500 mA, depending on the port design.
- **DC power jack (2.1 mm, center positive) for use with an external power source.** Recommended voltage range is 7-12 volts.
- **Low dropout regulators for 5V and 3.3V.** The five volt regulator provides up to 800 mA of current; the 3.3 volt regulator provides up to 50 mA. Connection pins are provided for both the 5V and 3.3V regulated outputs. You can use these pins to power low current components such as ultrasonic sensors or accelerometers.
- **Indicator LEDs for power, serial transmit and receive (labeled Tx and Rx), and digital pin 13 (labeled L).**
- **Six-pin in-circuit serial programming (ICSP) header.** This provides a standard connection with external programmers for the Atmel AVR microcontroller chips.
- **Six analog input/output (I/O) pins and 14 digital I/O pins.** The analog pins connect to an internal ten-bit analog-to-digital converter (ADC), letting you read voltages from sensors and other devices. All I/O



pins can be used as digital outputs, and can sink or source up to 40 mA.

- **Power pins to provide external access to the unregulated and regulated power supplies.**

Let me pause here to point out that the ATmega328 on the Uno board isn't an empty chip; it contains a small bootloader program for use with the Arduino development editor. The bootloader assists in the download process. You can add the bootloader yourself (instructions are on the *arduino.cc* website), or you can buy a replacement ATmega328 with the bootloader preinstalled.

Figure 3 shows the pin-out diagram of the 28-pin ATmega328. The labels on the inside of the chip are the primary function names for each of the pins. The labels outside in

The Chip-Only Arduino

While manufactured Arduino boards are hardly expensive, you can go even cheaper by using the Uno as a programmer. Once you've downloaded your sketch, remove the ATmega328 chip and transplant it into a solderless breadboard or other circuit. The chip runs under five volts (4.5V minimum, 5.5V maximum), and only needs a 16 MHz crystal and two 22 pF capacitors for operation. You can even do away with the caps if you use a 16 MHz three-pin resonator, and don't need the extra precision of a crystal oscillator.

Use an IC extractor tool to prevent damage to the ATmega328 pins when you remove it from the Arduino board. The tool grips the ends of the chip and allows you to pull it straight out of its socket.

There isn't even an absolute requirement that you use an ATmega328 with the Arduino bootloader preinstalled. You can use the Arduino development environment and download your programs directly into the chip. This

restores the Flash memory space previously taken up by the Arduino bootloader. It also avoids the several seconds' delay that occurs when the Arduino is first powered up; this delay is caused by the bootloader waiting to see if a new program is about to arrive.

Programming without the bootloader requires suitable hardware, such as the Atmel STK500, AVR-ISP, or a homebrew parallel port programmer. The Arduino Uno has a suitable in-circuit serial programming (ICSP) header already on it. Just attach the six-pin cable from your programmer to the ICSP header on the Arduino.

Just so you know, serial programming is just one method of burning software into an AVR microcontroller. Many stand-alone programmers like the STK500 also support what's known as *high-voltage programming* which permits resetting certain software fuse bits. These bits control special behaviors of the chip, and are documented in the AVR datasheets.

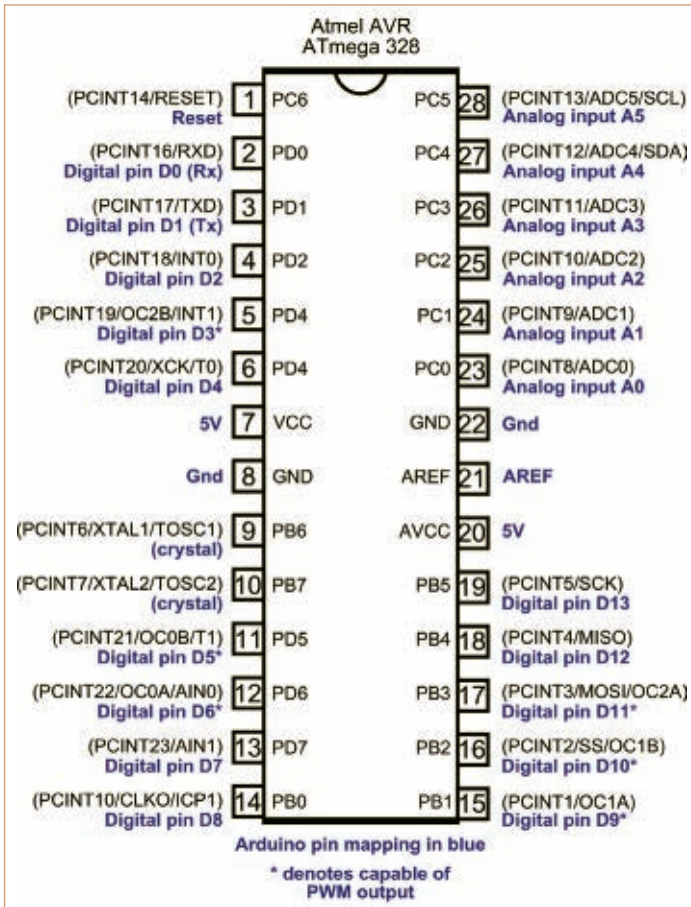


FIGURE 3. Pin-out diagram of the Atmel ATmega328 chip with the pin mapping to the Arduino I/O lines.

to worry about in typical Arduino programming, but it's nice to know what leads to where.

Writing and Downloading Programs

If you've used any kind of microcontroller, you know the process of programming it involves three steps: write the program; compile the program; and run the program (see **Figure 4**). The Arduino is no different, except that it refers to its programs as *sketches*.

Sketches are written in a programming language very similar to C. In fact, it is C (more accurately C++), but with some simplifications to make it easier for newcomers to master the system. If you've ever looked at a C/C++ program and felt your eyes glazing over because of the obtuse appearance of the code, you don't have to worry about that with the typical Arduino sketch. The Arduino is designed for beginners in mind, but still provides power and flexibility for more advanced users.

Taken indepth, the three steps of writing and downloading Arduino sketches are:

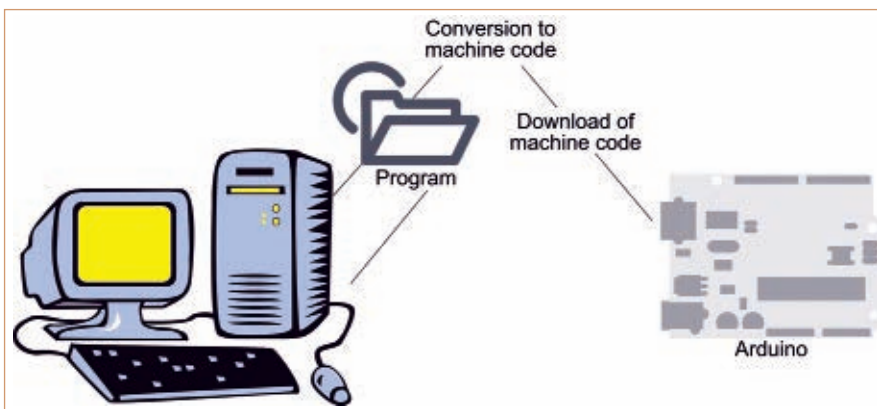
1. Develop your sketch on your PC. The Arduino comes with a Java-based IDE that includes a fully featured text editor. It supports syntax highlighting and coloring (different parts of code are shown in different colors), but doesn't give you popup hints or keyword suggestions — like Microsoft's Intellisense. If you're already familiar with another program editor like Eclipse or SEPY, you can use it instead. The file format for Arduino sketches is plain ASCII. (Even though SEPY is intended for programming ActionScript — the language used to create Adobe Flash applications — it inherently understands most of the C syntax used in Arduino sketches.)
 2. Once written, sketches must be compiled which in Arduino-land is referred to as *verifying*. During the compile/verify phase, any errors are flagged and noted at the bottom of the Arduino editor window. The compiling process includes building the sketch from component files. An Arduino sketch is not in itself completely compatible with C; for one thing, there's no *main()* function which is required to tell the compiler where the program is supposed to begin. In actuality, it's still there, under the hood. When you compile your sketch, the *main()* function is added to the program for you, along with some additional code.
 3. The compiled program is downloaded to the Arduino via a USB cable. The download process is automatic; the bootloader program residing in the Arduino detects when a new sketch is arriving. It performs the necessary steps of first erasing the old sketch in memory — if present — then accepting the new one. Once downloaded, the sketch starts automatically.
- When you download a compiled

parentheses are alternative uses — if any — for the pins.

For example, pin 9 — labeled PB6 (for Port B, bit 6) — is also used as a general-purpose I/O. In addition, it's used as one of two connection points for an external oscillator. As the Arduino uses a crystal oscillator connected to this pin — as well as pin PB7 — neither of these are available for use in your programs.

Also shown in **Figure 3** is pin mapping between the Arduino and the ATmega328. It's important to remember that the pin numbers are not the same between the two. Pin 12 on the ATmega328 is actually mapped to digital pin D6 on the Arduino. Pin mapping is not something you need

FIGURE 4. Programs (sketches) are developed on your PC, compiled to a machine-readable format, then downloaded to the Arduino via USB.



sketch to the Arduino, it is stored in 32K bytes of Flash memory inside the ATmega328. This memory is the same type used in solid-state USB drives, and has a lifetime of over 10,000 read/write cycles. Through the ATmega328, the Arduino also supports 1K bytes of electrically erasable non-volatile EEPROM (data survives after power-down) and 2K bytes of RAM. Data in RAM is volatile; it's lost when power is removed from the Arduino.

Arduino Architecture and Memory

Figure 5 shows a simplified block diagram of the ATmega328 used in the Arduino. In center stage is the central processing unit, or *CPU*. This piece is what runs your downloaded sketches, performing all the number crunching and other data processing tasks.

Feeding the CPU are the I/O lines, used to get data into and out of the chip. The I/O lines are the 20 analog and digital pins. Some of the pins are connected to special hardware within the ATmega328. For example, the six analog I/O lines go to the ADC, which translates an incoming voltage into any of 1,024 digital steps. Assuming a five volt incoming signal, the Arduino ADC provides a resolution of 4.9 millivolts per step.

The ATmega328 supports two external interrupts which are mapped to Arduino digital pins D2 and D3. Interrupts serve as a way to signal the CPU that a special event has taken place, without your sketch having to constantly check for it. Interrupts are set up in the Arduino IDE using the *attachInterrupt* programming statement. Along with this statement, you add the name of a function (I'll get to functions in a bit) that should run whenever the interrupt occurs.

There are also some blocks in the ATmega328 that are not exposed in the current versions of the Arduino IDE. There are no standard programming statements for them. An example is the analog comparator which triggers an interrupt when voltage on one comparator input equals or exceeds the voltage on another comparator input.

While current versions of the Arduino IDE don't have programming statements that directly support the analog compare function, that doesn't mean the Arduino isn't capable of using this feature on the ATmega chip. Remember, the Arduino programming language is based on C/C++ and links against the AVR Libc open source library which is a de facto standard for writing C programs on eight-bit Atmel AVR microcontrollers. Any function available in AVR Libc is also available on the Arduino. Or, let's put it this way: There's more to the Arduino than meets the eye, so don't be fooled by its apparent simplicity.

Anatomy of an Arduino Sketch

Part 1 of this series already touched on this topic, but it's worth repeating here: All Arduino sketches have at least two parts, named *setup()* and *loop()*. These are called *functions*, and they appear in the sketch like this:

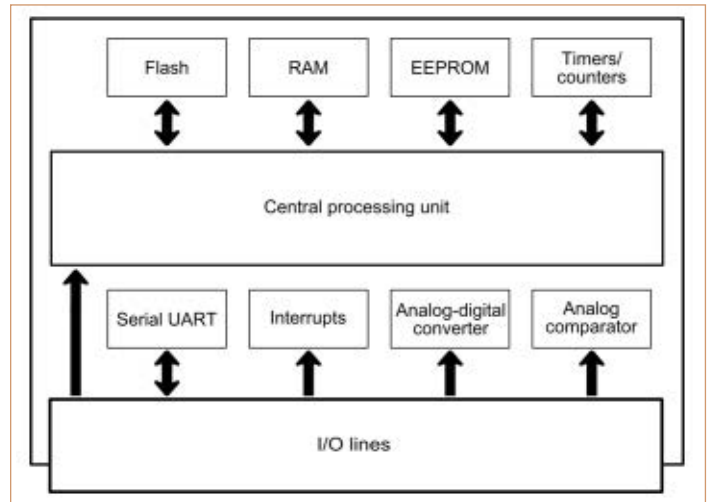


FIGURE 5. Simplified block diagram of the internals of the Atmel ATmega328 microcontroller.

```
void setup() {
}

void loop() {
}
```

The `()` parentheses are for any optional *arguments* (data to be used by the function) for use in the function. In the case of *setup* and *loop*, there are no arguments, but the parentheses have to be there just the same.

The `{ }` braces define the function itself. Code between the braces is construed as belonging to that function — the braces form what's referred to as a *code block*. There's no code shown here, so the braces are empty, but they have to be there just the same.

The *void* in front of both function names tells the compiler that the function doesn't return a value when it's finished processing. Other functions you might use (or create yourself) may return a value when they are done. The value can be used in another part of the sketch. We'll save this concept for a future article.

The *setup()* and *loop()* functions are required. Your program must have them or the IDE will report an error when you compile the sketch.

Arduino sketches may also have a *global declaration section* at the top. Among other things, the declaration is where you put variables for use by the whole program (see the following example). It's also a common place to tell the IDE that you wish to use an external library to extend the base functionality of the Arduino, and that programming code from that library should be included when your sketch is compiled.

Using libraries allows for convenient code re-use. The example code that follows uses the Servo library, which as its name suggests, provides an easy way to use R/C servo motors with the Arduino.

In Part 2, you saw a quick demonstration of operating the ArdBot's two servo motors. Let's review the core concepts behind that demo by looking at a simpler version; in this case, operating just one servo.

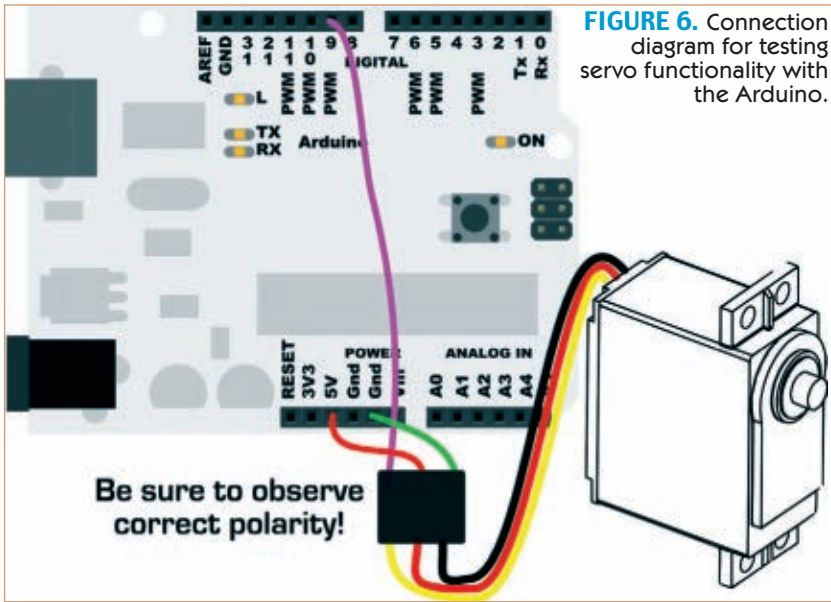


FIGURE 6. Connection diagram for testing servo functionality with the Arduino.

The program in **Code Example 1** swings the servo motor one direction, then the other, briefly pausing in between. You can use either an unmodified or modified (continuous rotation) servo to see the code in action. Refer to **Figure 6** for a diagram on hooking up the servo. Use a standard size (or smaller) analog servo; stay away from larger or digital servos, as they may draw too much current for the USB port on your computer to handle.

(In **Code Example 1**, text after the double slash `//` characters means a *comment*. It's for us humans. During the compiling phase, comments are ignored, as they are not part of the functionality of the sketch.)

The first line, `#include <Servo.h>`, tells the IDE that you want to use the Servo library which is a standard part of the Arduino IDE installation. (Other libraries may require a separate download, but they are used in the same way.) The name of the main Servo library file is `Servo.h`, so that is what's provided here.

The line `Servo myServo` creates, or "instantiates," a servo *object*; the functionality of this object is defined in the `Servo.h` library and its accompanying `Servo.cpp`

spaces, it may include only numbers, letters, and the `_` (underscore) character; and it can't be the same as any programming statements already defined for the Arduino.

The line `int delay = 1000` creates a data variable named *delay*. Variables are used to hold information for use throughout the sketch. The `int` tells the Arduino compiler that you wish to create an integer type variable which can store any whole number from -32,768 to 32,767. Other data types supported in the Arduino include *unsigned int* which holds values from 0 to 65,536, *byte* (holds 0 to 255), and *Boolean* (holds true or false).

The `setup()` function contains one statement, `myServo.attach(9)`. Here's what it all means:

- **myServo** is the name of the servo object that was defined earlier.
- **attach** is a *method* that you can use with the `myServo` object. Methods are actions that you use to control the behavior of objects. In this case, *attach* tells the Arduino that you have physically connected the servo to digital pin D9 and you want to activate it. A period separates the object name and method — `myServo.attach`.

Notice the `;` (semi-colon) at the end of the statement. It's a statement terminator. This tells the compiler that the statement is complete and to go on to the next line. The semi-colon is standard C programming syntax, but if you're used to a language like Basic — which simply uses hard returns to denote when a statement ends — the semi-colon business is bound to

```
#include <Servo.h>           // Use the Servo library, included with
                             // the Arduino IDE (version 0017 or later)

Servo myServo;               // Create a Servo object to control the servo
int delayTime = 2000;        // Delay period, in milliseconds

void setup()
{
  myServo.attach(9);         // Servo is connected to pin D9
}

void loop()
{
  myServo.write(0);           // Rotate servo to position 0
  delay(delayTime);           // Wait delay
  myServo.write(180);         // Rotate servo to position 180
  delay(delayTime);           // Wait again
}
```

CODE EXAMPLE 1

cause some initial troubles. You'll get used to it though, and before long you'll be adding semi-colons to everything you write — even grocery lists!

The `loop()` function contains the part of the sketch that is repeated over and over again until you download a new program or remove power from the Arduino. The function contains four lines.

- **`myServo.write(0)`** is another method using the `myServo` object. The `write` method instructs the servo to move all the way in one direction. When using a modified servo, this statement causes the motor to continually rotate in one direction.
- **`delay(delayTime)`** tells the Arduino to wait the period specified earlier in the `delayTime` variable which is 2,000 milliseconds (two seconds).
- The two statements are repeated again, this time with `myServo.write(180)` to make the servo go the other direction.

Before continuing, I want to mention an important note about capitalization of variables, objects, and statement names. Like all languages based on C, these names are case sensitive, meaning `myServo` is distinctly different from `myservo`, `MYSERVO`, and other variations. If you try to use

```
myservo.attach(9);
```

(note the lower-case *s*) when you've defined the object as `myServo`, the Arduino IDE will report an error — "myservo not declared in this scope." If you get this error, double-check your capitals.

More Experiments with Servo Objects

The Servo class provides a number of methods that can be used on its objects. I recommend you check out the documentation for the Servo library on the arduino.cc website, but here are the principle ones you should know about:

- **`attach`** connects a servo object to a specific pin of the Arduino. You can use any pin.
- **`detach`** removes the servo object, effectively disabling the servo and removing its power.
- **`write`** specifies where to position the servo. The method accepts several forms of values. A value of 0 to 180 denotes degrees; this positions the shaft of the motor to a corresponding angle. (When used with modified servos, 0 and 180 make the motor turn one direction or the other; 90 makes the

```
void setup()
{
  myServo.write(180);    // Start at 180 degrees instead of 0
  myServo.attach(9);
}
```

CODE EXAMPLE 2

```
void loop()
{
  myServo.attach(9);    // Attach and apply power
  myServo.write(0);     // Position servo
  delay(delayTime);     // Allow transit time
  myServo.detach();     // Detach and remove power
  delay(4000);          // Wait 4 seconds
  myServo.attach(9);    // Re-attach and apply power
  myServo.write(180);   // Move servo to other end
  delay(delayTime);     // Allow transit time
  myServo.detach();     // Detach again
}
```

CODE EXAMPLE 3

motor stop.) Values from 544 to 2400 are treated as microseconds and position the servo by generating pulses of the specified duration. Typical servo specs are 1,000 to 2,000 microseconds for a standard 60 degree arc.

- **`writeMicroseconds`** specifically indicates you wish to use microseconds to control the servo position.
- **`read`** returns the last specified position of the servo in degrees.

One technique to try is writing a position to the servo *before* calling the `attach` method — attaching the servo is what gives it power. When you create a new Servo object, its position is automatically given a default of 0. By setting a position first, then attaching the servo, you can have it start at a position other than 0 degrees. See **Code Example 2**.

There's also no absolute requirement that you use the `attach` method in the `setup()` function. You can place it in the `loop()` function and use the `detach` method to remove power to the servo. **Code Example 3** demonstrates sweeping the servo right and left, while stopping it (actually turning it off) for four seconds in between. The action is a bit easier to see when using a servo modified for continuous rotation.

Sources

Arduino
www.arduino.cc
 Prefabricated ArdBot body pieces with all construction hardware.

Partial list of Arduino resellers:

Budget Robotics
www.budgetrobotics.com

AdaFruit
www.adafruit.com

HVW Tech
www.hvwtech.com

Jameco
www.jameco.com

Pololu
www.pololu.com

Robotshop
www.robotshop.com

Solarbotics
www.solarbotics.com

Sparkfun
www.sparkfun.com

Arduino circuits and sketches submitted by users:

Fritzing
www.fritzing.com

Detaching the servo will prevent it from buzzing, or if using a servo modified for continuous rotation, will stop it from slowly “creeping” when you set its position to 0 (stop). Since the servo is not being powered, it also saves battery juice when your Arduino and servos are mounted on a mobile robot.

(Okay, detaching to remove power probably won’t work with digital servos. Detaching stops the Arduino from sending pulses to the servo, which on analog models — what most people use — effectively shuts them off. Digital servos will continue to hold position even when its pulses go missing. That’s what they are intended to do. So, the above really applies to your typical, everyday, garden variety analog servo.)

Creating Your Own Functions

The flexibility of any programming language — Arduino included — comes in the ways you can develop reusable code, such as creating user-defined functions. To create a user-defined function, you give it a unique name and place the code you want inside a pair of brace characters, like so:

```
void forward() {
  myServo.write(0);
  delay(delayTime);
}
```

All user-defined functions must indicate the kind of

data they return (for use elsewhere in the sketch). If the function doesn’t return any data, you use *void* instead. You must also include parentheses to enclose any parameters that may be provided for use in the function. In the case of the *forward* user-defined function, there are no parameters, but remember you need the (and) characters just the same.

That defines the function; you only need to *call* it from elsewhere in your sketch to use it. Just type its name, followed by a semi-colon, to mark the end of the statement line:

```
forward();
```

See **Listing 1** for a full demonstration of an Arduino sketch that runs a servo forward and backward, then briefly stops it using the detach method. Recall the effect of the sketch is most easily seen when using a servo modified for continuous rotation, as is the case for a robot like the ArdBot that uses continuous rotation servos to propel it across the floor.

Finally, a Word About IDE Versions

The Arduino IDE and the standard programming statements and libraries often undergo changes with each new version. The Servo library as detailed here was introduced in version 0017 of the Arduino IDE. As of this writing, we’re up to version 0021.

If you already have an installed version of the IDE and it’s old, you’ll want to fetch the newest version. You can keep multiple versions of the Arduino IDE on your computer, and even switch between them as needed — though that should seldom be required.

The ArdBot project requires version 0017 or later. I’ve tested everything on version 0019, plus the latest build (0021) just to make sure everything still works as it should. The Arduino IDE is set to always check for the latest updates. If you open the IDE and it tells you a new update is ready, download and install it, and be sure to take a look at the readme file for the latest changes.

In future installments, you’ll be integrating what you’ve learned here with the ArdBot robot, including writing your own customized servo motor control functions, plus adding sensors to your ArdBot to make it detect and avoid obstacles, and more. **SV**

Listing 1

```
#include <Servo.h>

Servo myServo;           // Create Servo object
int delayTime = 2000;    // Standard delay period (2 secs)
const int servoPin = 9;  // Use pin D9 for the servo

void setup() {           // Empty setup
}

void loop() {            // Repeat these steps
  forward();             // Call forward, reverse, servoStop
  reverse();             // user-defined functions
  servoStop();
  delay(3000);
}

void forward() {         // Attach servo, go forward
  myServo.attach(servoPin); // for delay period
  myServo.write(0);
  delay(delayTime);
  myServo.detach();      // Detatch servo when done
}

void reverse() {         // Do same for other direction
  myServo.attach(servoPin);
  myServo.write(180);
  delay(delayTime);
  myServo.detach();
}

void servoStop() {       // Stop the servo by detaching
  myServo.detach();
}
```

Gordon McComb can be reached at
arduino@robotoid.com.

The *SERVO* Webstore

Attention Subscribers ask about your discount on prices marked with an *

CD-ROM SPECIALS



6 CD-ROMs & Hat Special
Only \$ 129.95 or \$24.95 each.
www.servomagazine.com

The Amateur Scientist 3.0 The Complete Collection by Bright Science, LLC

There are 1,000 projects on this CD, not to mention the additional technical info and bonus features. It doesn't matter if you're a complete novice looking to do your first science fair project or a super tech-head gadget freak; there are enough projects on the single CD-ROM to keep you and 50 of your friends busy for a lifetime!

Reg \$26.95 Sale Price \$23.95



ROBOTICS

PIC Robotics by John Iovine

Here's everything the robotics hobbyist needs to harness the power of the PICMicro MCU!

In this heavily-illustrated resource, author John Iovine provides plans and complete parts lists for 11 easy-to-build robots each with a PICMicro "brain." The expertly written coverage of the PIC Basic Computer makes programming a snap – and lots of fun.

\$24.95



FIRST Robots: Rack 'N' Roll: Behind the Design

by Vince Wilczynski,
Stephanie Slezyski

More than 750 photographs!

The second annual book highlighting the creativity and process behind 30 winning robot designs from the 18th annual international FIRST Robotics Competition. The FIRST organization, founded by Dean Kamen

(inventor of the Segway), promotes education in the sciences, technology, and engineering.

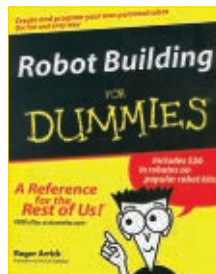
Reg \$39.95



Robot Building for Dummies by Roger Arrick / Nancy Stevenson

Discover what robots can do and how they work. Find out how to build your own robot and program it to perform tasks. Ready to enter the robot world? This book is your passport! It walks you through building your very own little metal assistant from a kit, dressing it up, giving it a brain, programming it to do things, even making it talk. Along the way, you'll gather some tidbits about robot history, enthusiasts' groups, and more.

\$24.95



Build Your Own Humanoid Robots

by Karl Williams

GREAT 'DROIDS, INDEED!

This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting projects, each costing less than \$300. Together, they form the essential ingredients for making your own humanoid robot. **\$24.95***



Robot Programmer's Bonanza by

John Blankenship,
Samuel Mishal

The first hands-on programming guide for today's robot hobbyist!

Get ready to reach into your programming toolbox and control a robot like never before! *Robot Programmer's Bonanza* is the one-stop guide for everyone from robot novices to advanced hobbyists who are ready to go beyond just building robots and start programming them to perform useful tasks.

\$29.95



Robotics Demystified

by Edwin Wise

YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS!

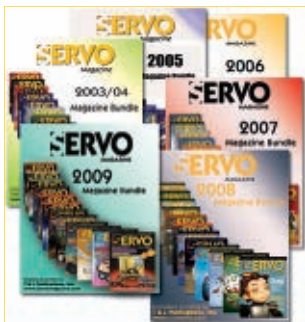
Now anyone with an interest in robotics can gain a deeper understanding – without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **\$19.95**



**We accept VISA, MC, AMEX,
and DISCOVER
Prices do not include shipping and
may be subject to change.**

To order call 1-800-783-4624

SERVO Magazine Bundles



Published by T & L Publications, Inc.

\$57
per bundle

Save \$10
off the
normal
price!!

Now you can get one year's worth of all your favorite articles from *SERVO Magazine* in a convenient bundle of print copies. Available for years 04, 05, 06, 07, 08, and 09.

Kickin' Bot

by Grant Imahara

Enter the arena of the metal gladiators!

Do you have what it takes to build a battle-ready robot? You do now! Here are the plans, step-by-step directions, and expert advice that will put you in competition — while you have a heck of a lot of fun getting there. Grant Imahara, the creator of the popular BattleBot Deadblow, shares everything he's learned about robot design, tools, and techniques for metal working and the parts you need and where to get them.

\$24.95

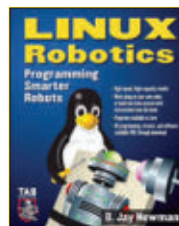


Linux Robotics

by D. Jay Newman

If you want your robot to have more brains than microcontrollers can deliver — if you want a truly intelligent, high-capability robot — everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need.

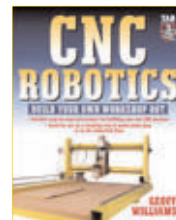
\$34.95



CNC Robotics

by Geoff Williams

Here's the FIRST book to offer step-by-step guidelines that walk the reader through the entire process of building a CNC (Computer Numerical Control) machine from start to finish. Using inexpensive, off-the-shelf parts, readers can build CNC machines with true industrial shop applications such as machining, routing, and cutting — at a fraction of what it would cost to purchase one. Great for anyone who wants to automate a task in their home shop or small business. **\$34.95**



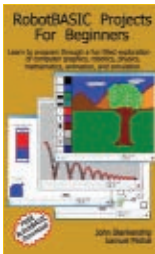
SPECIAL OFFERS

RobotBASIC Projects For Beginners

by John Blankenship, Samuel Mishal

If you want to learn how to program, this is the book for you. Most texts on programming offer dry, boring examples that are difficult to follow. In this book, a wide variety of interesting and relevant subjects are explored using a problem-solving methodology that develops logical thinking skills while making learning fun. RobotBASIC is an easy-to-use computer language available for any Windows-based PC and is used throughout the text.

Reg. Price \$14.95 Sale Price \$9.95



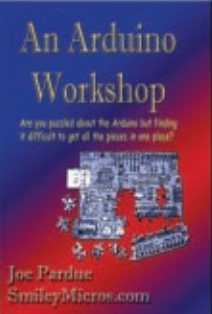
Technology Education Package for Everyone Starting in Electronics

This lab — from the good people at GSS Tech Ed — will show you 40 of the most simple and interesting experiments and lessons you have ever seen on a solderless circuit board. As you do each experiment, you learn how basic components work in a circuit. Along with the purchase of the lab, you will receive a special password to access the fantastic online interactive software to help you fully understand all the electronic principles. For a complete product description and sample software, please visit our website.

Regular Price \$79.95

Subscriber's Price \$75.95

SPECIAL OFFERS



Book \$44.95

Puzzled by the Arduino?

Based on the *Nuts & Volts* Smileys Workshop, this set gives you all the pieces you need!

Book and Kit Combo \$124.95



Kit \$84.95

For more info on this and other great combos, please visit: <http://store.nutsvolts.com>



Combo Price \$175.95

Enter the world of PICs & Programming with this great combo!

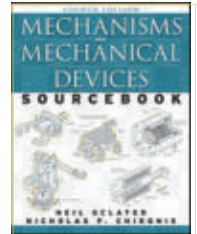




For complete details visit our webstore @ www.nutsvolts.com

Mechanisms and Mechanical Devices Sourcebook

by Neil Sclater, Nicholas Chironis
Over 2,000 drawings make this sourcebook a gold mine of information for learning and innovating in mechanical design. Overviews of robotics, rapid prototyping, MEMS, and nanotechnology will get you up to speed on these cutting-edge technologies. Easy-to-read tutorial chapters on the basics of mechanisms and motion control will introduce those subjects to you. **Reg \$89.95 Sale Price \$69.95**



Forbidden LEGO

by Ulrik Pilegaard / Mike Dooley
Forbidden LEGO introduces you to the type of free-style building that LEGO's master builders do for fun in the back room. Using LEGO bricks in combination with common household materials (from rubber bands and glue to plastic spoons and ping-pong balls) along with some very unorthodox building techniques, you'll learn to create working models that LEGO would never endorse. **Reg \$24.95 Sale Price \$19.95**



PROJECTS

The SERVO Buddy Kit



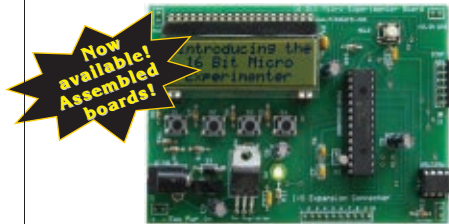
An inexpensive circuit you can build to control a servo without a microcontroller.



For more information, please check out the May 2008 issue or go to the **SERVO** websore.

Includes an article reprint.
Subscriber's Price \$39.55
Non-Subscriber's Price \$43.95

16-Bit Micro Experimenter Board



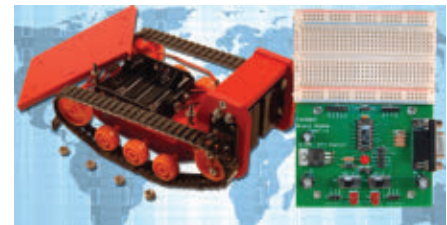
Ready to move on from eight-bit to 16-bit microcontrollers? Well, you're in luck!

In the December 2009 *Nuts & Volts* issue, you're introduced to the 16-Bit Micro Experimenter.

The kit comes with a CD-ROM that contains details on assembly, operation, as well as an assortment of ready-made applications. New applications will be added in upcoming months.

Subscriber's Price \$55.95
Non-Subscriber's Price \$59.95

Tankbot Kit & Brain Alpha Kit



Tankbot/Brain Alpha
originally by Ron Hackett

A series filled with projects and experiments to challenge you through your learning process while you grow your fully expandable Brain Alpha PCB! The brain is a PICAXE-14A!

For more info & pictures, visit the *SERVO* Webstore. Tankbot and the Brain Alpha Kit can be purchased separately.

Combo Price \$138.95

Twin Tweaks

THIS MONTH:
Cirque Du Robot



THE
SMARTSENSOR
LITE FROM
CATCAN
CREATIVE.



A lot of robotics kits make for great hacking platforms, and to get the most out of an expandable kit you need versatile sensors. This month, we have the opportunity to demonstrate both – an expandable platform with the LEGO NXT kit, and a multitasking sensor with the SmartSensor Lite from CATCAN Creative, Inc.

The SmartSensor Lite is equipped with accelerometers, a gyroscope, and even a temperature sensor. We thought that such a dynamic sensor would be perfect for some gravity-defying stunts in our very own Cirque du Robot.

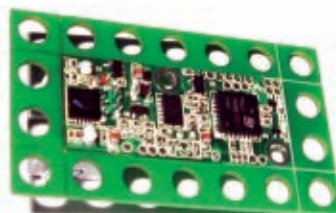
Opening Up the Magic Box

CATCAN Creative is a Taiwanese company dedicated to designing sophisticated sensors and servos for hobbyists and educational use. One of their products is the SmartSensor Lite. The SmartSensor Lite is indeed compact and lightweight, and the telltale pattern of holes around the border of the PCB indicates that the unit is designed to

work with the LEGO Mindstorms NXT kit. The folks at CATCAN sought to create an easy to use sensor capable of capturing all sorts of motion data. The well equipped sensor module used a tri-axial accelerometer that takes measurements of the traditional Tait-Bryan angles – pitch, yaw, and roll. A gyroscope measures angular velocity about the yaw axis. The unit even comes with a temperature sensor. The temperature sensor might seem like a random addition, but with a stated range of -40° C to 125° C, we're sure that intrepid hackers will figure out a way to put it to good use. The specs on all of the sensors are available on the infinitely helpful CATCAN website (www.catcan.com.tw).

The website is in Taiwanese, but thanks to Google Translate you don't need to brush up on your language skills before diving in. In addition to providing all of the sensor specs, the CATCAN website has instruction manuals, software guides, and amusing videos highlighting the capabilities of the SmartSensor Lite. We found one of the videos particularly inspiring. It was a video of a LEGO NXT robot balancing on two wheels, much like a Segway. The bot was mercilessly targeted by flying wheels and nudging feet, but it always gracefully maintained its balance. We thought that trying to replicate such behavior would be a good challenge, especially because such graceful balance is not something that the stock NXT kit could easily achieve.

GETTING ACQUAINTED WITH THE SMARTSENSOR LITE.



Reinventing the Mindstorms

We've been lucky enough to make several forays into robotics competitions using a fleet of LEGO Mindstorms robots. From a whole family of robots named Gog designed for the Science Olympiad competition to a group of contenders at the 2003 event sponsored by the Phoenix Area Robotics Experimenters, we've been consistently impressed by the versatility of the Mindstorms kit. But despite such an illustrious history with this kit, our experience with the NXT has been relatively minor – making only a supporting appearance in the kinetic sculptures of the COSMOS program (see the January '10 issue). We were eager to finally get firsthand experience with the new and improved NXT kit, and we had high hopes that it would be as intuitive as its RCX brick-driven predecessor. We were particularly interested to see how accommodating the kit would be to a sophisticated third party device like the SmartSensor Lite.

We speedily assembled a two wheeled base, leaving room to fasten the SmartSensor Lite to the side of the bot. As far as mechanical implementation, the SmartSensor Lite couldn't have been more intuitive. The holes around the border of the PCB align perfectly with the NXT structural components, and the blue connector axles coupled with the orange doodads seem destined to fasten the module securely to your LEGO robot. The SmartSensor Lite wires up to the NXT brick with the same RJ12 connectors as the rest of the kit, making integration as seamless as possible.

With the mechanical implementation completed, we arrived at the real challenge – programming. The NXT programming felt pleasantly familiar. The block oriented programming retained the accessibility of the first generation Mindstorms interface, and the new and very apparent influence of the folks at National Instruments has added the sophistication and versatility of LabView. To get ourselves oriented to the programming, we put together a quick program that moved our drive base around as it reclined in its natural position.

To put the CATCAN sensor to work, we had to import a special block into the NXT programming environment. An instruction manual from the CATCAN website walks you through the process step by step, which is basically a matter of choosing to import a block from the NXT menu and finding the CATCAN block in the correct folder downloaded from the website. The CATCAN block then appropriately appears under the "Advanced" blocks flyout in the menu, and can be



BUILDING UP A DRIVING BASE WITH NXT.

used like any other block.

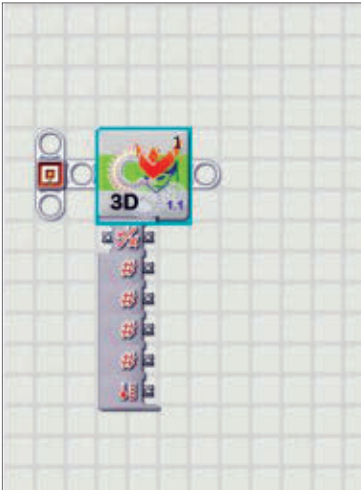
The SmartSensor Lite can be used on any one of the NXT's sensor ports, and the block contains five possible outputs: pitch, roll, heading, gyro, and temperature. One of the sample programs that the instruction manual walks you through is a perfect debugging program. The simple program takes an output from the SmartSensor Lite and displays it on the NXT screen. We used this program to check the orientation of the sensor, and to make sure we knew what the value was corresponding to the upright position and the value's characteristic of tipping over in each direction.

A raw data block can also be imported into the NXT programming environment. The raw block can output the raw data from the accelerometer or gyro, for all those folks

THE NXT IN ITS NATURAL RESTING STATE.



Twin Tweaks ...



THE SMARTSENSOR LITE NXT PROGRAMMING BLOCK.

who like their data pure and unadulterated and not in handy degree form.

A Robotic Saltimbanque

Implementing the SmartSensor Lite is a great exercise in connecting the logic of the programming to the real world system. After confirming that the heading reading changed as the robot leaned back and forth, we had to connect that with the

response that we thought would prevent the robot from falling. The most basic solution would be for the robot to move backwards as it fell backwards and move forward as it fell forward. The trick would be for the robot to move at the right speed and by the right amount when it knew it was tipping. The best way to determine tipping would be with the gyroscope reading – if the angular velocity became too great, the robot would move to correct it.

BIPEDAL BALANCING IS A NOUVELLE EXPERIENCE FOR AN NXT BOT.



At first, we wrote a simple program in the basic NXT interface. We used a “compare” block to compare the output from the gyroscope to a number that would indicate the tipping point of the robot. Because the default output of the gyro is positive in the counterclockwise direction, this meant a positive value corresponded to falling backwards. So if the comparison was true – the gyro reading was greater than the tipping point – the robot should move backwards to compensate. If the reading was false, the robot should move forward to compensate. We tested the robot by leaning it back and forth by hand, and the motors appeared to react appropriately.

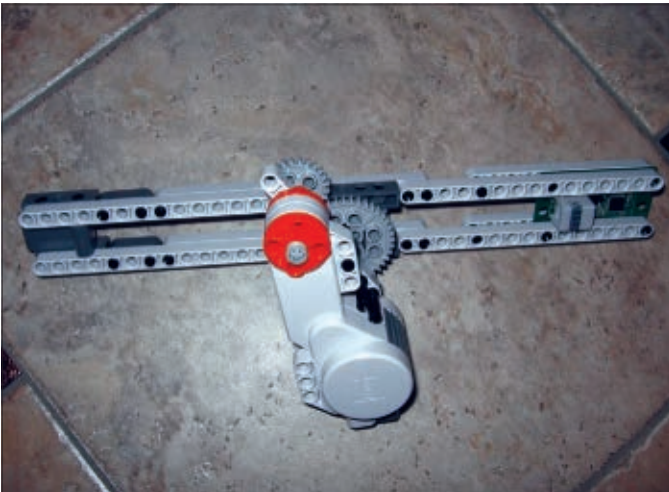
Even after an encouraging test, though, our robot wouldn't stay upright. It wiggled back and forth, and could even avoid a pratfall for a second or two while doing its best hip gyrating Elvis impression, but our balancer was never able to maintain its balance. No matter what we tried, our two wheeler was like the IFC Channel – always slightly off.

We took another look at our far too simple program and spotted some problems. The compare blocks in particular struck us as troublesome. Because of how we defined our tipping point, a false reading from the compare block could correspond to a stable state. But a false reading also came back as the robot was falling forward. Basically, our simple program was unable to recognize a stable state and thus couldn't keep its balance. Of course, a more sophisticated program would want to ramp down the acceleration as the robot approached the stable state, and it would be better to base the motor output decisions on more than a single instantaneous reading of angular velocity.

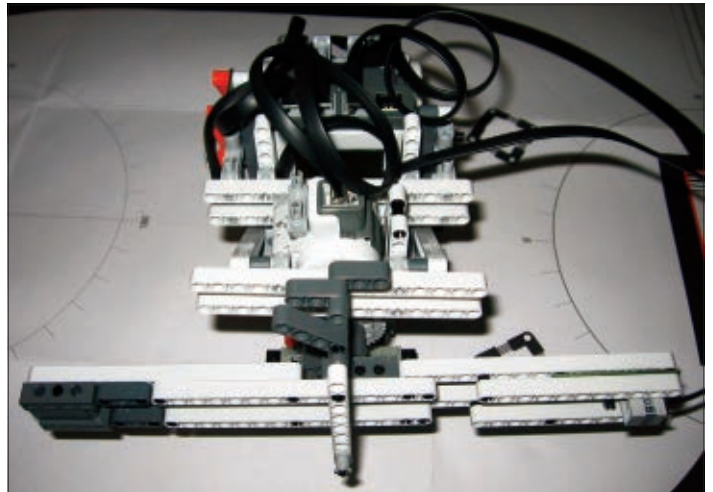
The SmartSensor Lite is intended for static measurement, but the balancing robot should still be theoretically possible. Indeed, the videographic evidence was taunting us on the website. The robot on the web, however, had an advantage – it was programmed in Robot C and used a Kalman filter and PID control. That is indeed one of the charms of the LEGO NXT kit – it can be used with a number of programming languages, like Robot C, NXC, and others. However, we wanted to see what the SmartSensor Lite was capable of given the limitations of the traditional NXT programming.

Solving the Mystère of Balance

We wanted to implement the SmartSensor Lite in a way that would still take advantage of its measurement capabilities while being programmed with the NXT software. We wanted to do something acrobatic, and inspiration hit us like a flying Wallenda. The gentle sway of a balancing rod is far more suited for a sensor module accustomed to static measurements, but it still allows for stunning feats of balance and poise. Our plan was to equip our robot with a balancing rod, but instead of walking a tightrope we would have it negotiate some uneven ramps around a line following track.



A BALANCING ARM MECHANISM.



IMPLEMENTING THE SMARTSENSOR.

To actually help the robot maintain its balance by shifting the center of mass, the balancing arm would need to have some weight to it. To ensure that the arm would still be able to move quickly to accommodate uneven terrain while having enough torque to swing a weighty beam, we used a 1 to 2.5 gear ratio. The SmartSensor was mounted so that the pitch was the relevant measurement. We weighted the ends of the balancing rod with old school film cans filled with pennies.

Programming the balancing rod is once again an exercise in relating the physical operation of the mechanism to the logic of the programming. The SmartSensor in particular makes use of the loops and branching programs that come so naturally to the NXT programming. Logically, we knew that a range of readings by the accelerometer would correspond to a stable zone where the robot need not deign to adjust the balancing rod. If the reading indicated a danger of tipping over, the balancing rod would correct and move to a level position. We used range blocks and a branching pattern to program a response for the stable zone, tipping to the left and tipping to the right.

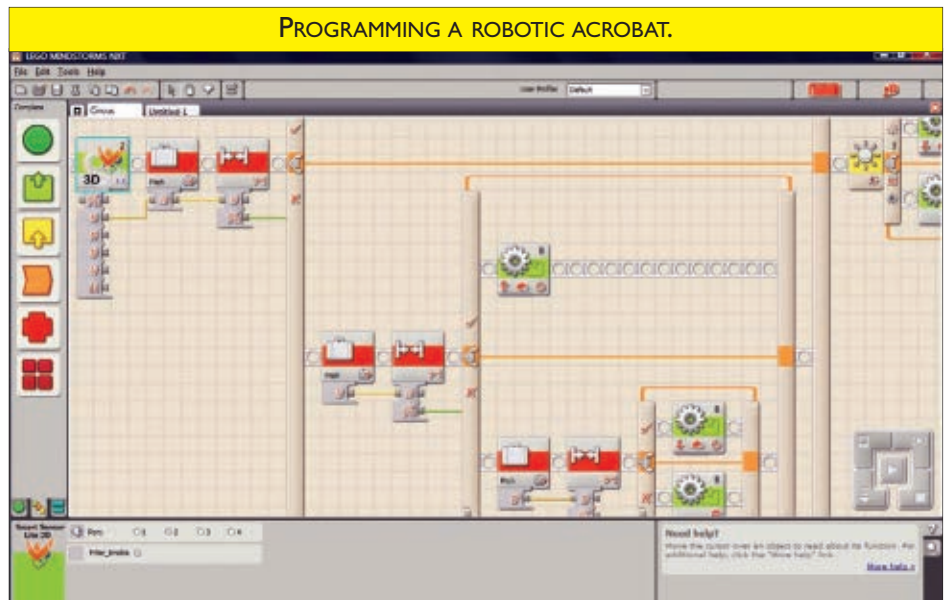
Coupling this with a line following program was a simple matter of adding another set of blocks outside of the main balancing program. Our initial tests with the balancing rod left much room for improvement. At first, we made some silly oversights, and the balancing rod moved in the opposite direction than we wanted which greatly aggravated the instability of our robot. By once again checking the positive direction of the motor, the switch in direction because of the gear ratio, and the positive direction for the SmartSensor reading, we sorted out the basic behavior of the balancing rod.

That, however, was not the end of our tribulations. The balancing rod tended to oscillate instead of chill in the stable zone. We troubleshot by displaying the pitch data on the screen, and we discovered that the sensor was too responsive to the vibrational vagaries of the balancing rod. We installed the SmartSensor Lite on one end of the balancing rod, and the weighted ends, long moment arm, and lack of smooth ramp down in acceleration caused the ends of the beam to bounce a bit even after the motor had made its correction. By increasing the size of our stable zone, we stopped the oscillations, though this would also be a great opportunity for a more elegant programming solution with a ramp down in acceleration as the stable zone approached.

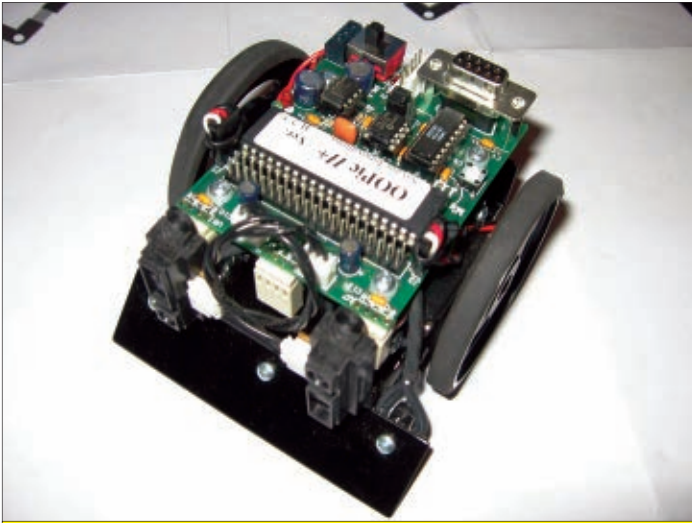
Like a Big Glittery Banana Peel

After we finished troubleshooting, we wanted to test our bot's balance in an acrobatic spectacle. Of course the

PROGRAMMING A ROBOTIC ACROBAT.



Twin Tweaks ...



HOUSE ROBOT MARK III.

most entertaining way to test the capabilities of our projects is through a little friendly competition. Our dad's Mark III robot is as ready to perform as a fixture at a Las Vegas hotel, even though the low profile Sumo robot is far more used to flat terrain than the uneven track of the Cirque du Robot.

To make our obstacle course, we started with the LEGO

THE STOCKY MARK III DOES NOT APPRECIATE OBSTACLES.



line following track and put ramps to one side of the line. We wanted one wheel of the robot to be picked up by the ramp, shifting the center of mass beyond the wheelbase until the robot tipped over in maladroitness. To make sure that our ramps were sufficiently devious, we tested them on our balancing robot sans balancing rod. Our line following program toodled along just fine until it hit a ramp, but then our top heavy bot tipped over like a drunk elephant on a tightrope. Oh the Zumanity!

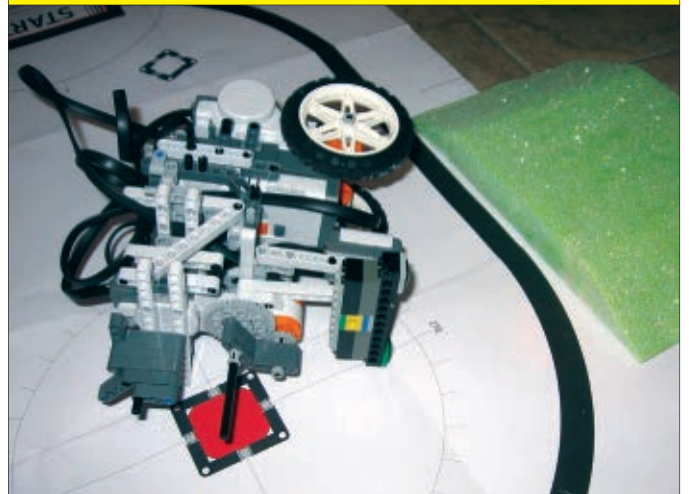
Likely buoyed by the embarrassing performance of its rival, the Mark III was ready to take a run around the track. The Mark III has a fairly low center of gravity, as is characteristic of a robot designed for Sumo competition. The robot also has a narrow wheelbase, and a ramp of decent height would knock it over. The ramps on the track were indeed high enough, and the Mark III pratfalled like a malchanceux trapéziste.

Now that the suspense had been deftly cultivated, we reattached the balancing arm to our NXT bot for another run. The line following began without incident, and as the robot ascended the ramp our balancing arm swung to life. Like a well trained performer, the SmartSensor Lite directed the arm back to a level position, and the robot was able to conquer the ramp. The obstacle didn't even distract the bot from the line following track. Thus, the SmartSensor Lite and NXT programming was vindicated by the less rapid balancing act of the balancing beam, which demonstrated that the NXT kit is accommodating to sophisticated third party sensors and the CATCAN sensor adds useful motion measuring capability to the NXT.

C'est la Fin de l'Article

Even though our project focused more on a plug and play application of the CATCAN device, the charms of the SmartSensor Lite for advanced roboticists are obvious. The module can create sophisticated robot behaviors (like balancing on two wheels) implementing more advanced

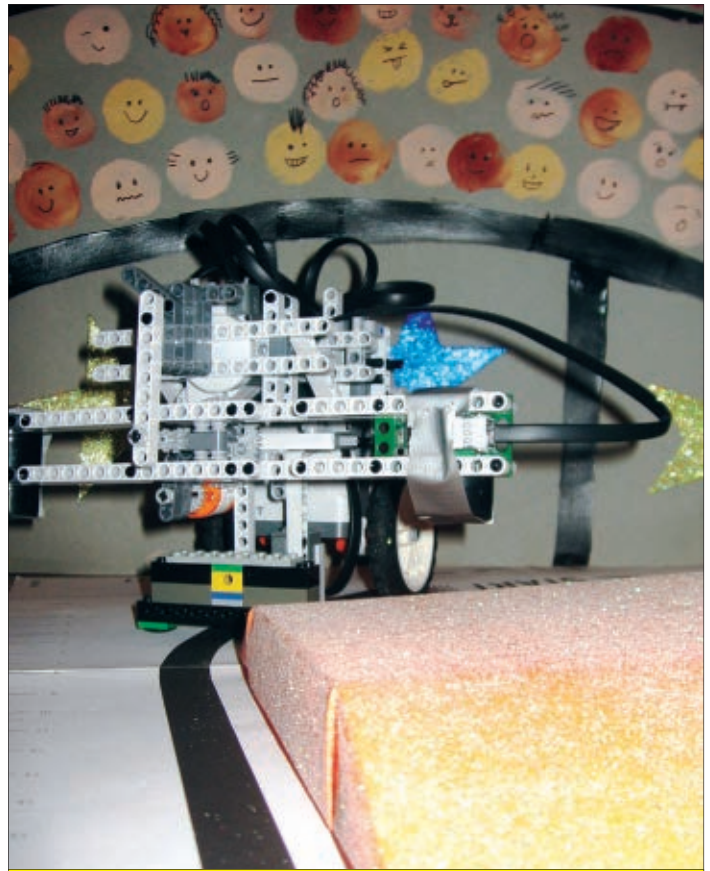
WITHOUT A BALANCING ROD, THE BOT IS A FALLING WALLENDIA.



programming techniques using NXC and Robot C. The website even provides a Virtual Instrument that can be used with National Instruments' LabView software. The sensor module is also compatible with Arduino and any microcontroller that can interface with I²C. To interface with Arduino, users need to get a special adapter cable; one that can readily be made by hacking separate female connectors onto the component wires of an RJ12 cable. The manual gives instructions on how to wire the SmartSensor to the Arduino, using the Deumilanove board as an example. The manual also includes the I²C register table for the SmartSensor data, and it offers some sample code for reading the sensor output with the Arduino.

The folks at CATCAN also have plenty of other devices that would appeal to advanced hackers. The SmartSensor Pro is equipped with all of the goodies of its Lite cousin, plus embellishments like triaxial gyro instruments, data logging capabilities, and a USB or UART interface — perfect for more advanced projects that need an easy to use motion sensor. Such a project might also require some servos, and CATCAN offers a SmartServo with built-in PID control and the ability to report such useful information as coordinates, temperature, and load voltage. To work up to such involved projects, a roboticist has got to start somewhere.

In the world of robotics kits, there are numerous products targeted at a novice crowd, and plenty of options for expert hackers. One of the most important ways to keep newcomers involved is to have kits that can transition from beginner's projects to advanced tinkering. We think the SmartSensor Lite can serve as a perfect stepping stone from beginner projects with NXT to more advanced robots using NXT and beyond. In particular, we think the SmartSensor Lite is a great way to add a level of abstraction to the sensor systems of the NXT kit. Most of the NXT sensors are easy to logically understand at an intuitive level. Touch sensors are buttons that can tell when they are pressed or not; light sensors can register the difference between a white surface and the black line following track. However, the SmartSensor Lite, even though it measures something tangible — angular position and angular velocity — it does so by spitting out a number instead of an easy to digest true or false. Freshman roboticists need to connect that number to what the sensor is reading, and



APPROACHING THE OBSTACLE.

manipulate it in the program so that the robot responds with the proper behavior.

The motion sensing capabilities of the SmartSensor Lite pose a great challenge to hackers. Sure, it can be implemented with simpler projects like a balancing rod, but the real brass ring is the bipedal balancing demonstrated on the website. By creating a sensor that can be used in conjunction with a variety of different kits and for a variety of different tasks, the folks at CATCAN offer a module that can be used by both beginners and experts alike, and, more importantly, can help those beginning tinkerers of today become the expert hackers of tomorrow. **SV**

WOWING THE CROWD.



SPECIAL THANKS TO

Henry Hung from CATCAN

Trish McDonell from
National Instruments

**RECOMMENDED
WEBSITES**

www.catcan.com.tw



Then and NOW

ROBOT HANDS

b y T o m C a r r o l l

Over the years, I've discussed all sorts of robot manipulators and arms, but most of these have been related to industrial applications. The 'end effectors' of most factory robots are quite varied in form and function but are usually specific to a particular task. There are many types of welding robots including spot welders and cutting heads, but these end effectors are not applicable to any other sort of task. Pick and place robots may have several types of 'hands' or 'claws' such as the popular parallel-jaw claw, but these devices might have a difficult time manipulating a piece of glass. SCARA robots are great for precision x-y tasks such as populating circuit boards but their end effectors might be designed for a narrow range of objects to be inserted such as resistors, caps, or diodes. Human hands may tire and get injured in factory jobs but they are — without a doubt — the most universal hand that has ever worked in a factory. Fingered robot hands and similar manipulators are becoming the top priority for robot researchers world-wide. The DARPA sponsored Autonomous Robot Manipulation (ARM) program is a good example as they want to move robotic manipulation "from an art to a science."

Last month, I featured the European space robot, Justin, and compared it with NASA's Robonaut. These robots use human-like hands to assist or replace astronauts in difficult space-borne operations. In down-to-earth applications, dexterous manipulators have been designed by many university laboratories through the years to provide amputees with the ability to grasp and manipulate objects in their daily lives. All robots do not require complex five-fingered hands in order to grasp and handle objects. Willow Garage's PR-2 shown in **Figure 1** does a fine job with a two-fingered 'pincer-type' claw, but a humanoid style five-fingered hand is the most universal end-effector. It is also the most complex and difficult to build.

Figure 2 is a patent drawing of a hand designed in Germany in 1916 for amputees. Designed

by William Carnes, the hand is very sophisticated for its time. Notice that the Carnes hand's fingers are driven by a single worm gear though the thumb appears to be stationary.

Anthropomorphic Robot Hands



'Anthropomorphic' is a mouthful of letters that literally means 'man-formed' or 'in the form of man.' For those who strive for the most realistic humanoid robot, hands in the form of human hands are the way to go. Think about it for a moment. If it weren't for our opposable thumb, much of our hand's uniqueness would be lost. Without those thumbs, much of our manipulation

FIGURE 1. Willow Garage's PR-2 with pincer claw hand.

FIGURE 3. Robot hand and fingers using bendable tubing (scanned copy of whole page of drawings).

capability of objects would require two hands. Spend a few minutes trying to pick up or manipulate objects without using your thumbs. Even though the thumb lacks a degree of freedom in motion (as compared to our fingers) and is not particularly maneuverable in its own right, it adds so much to our hand's capabilities. However, our hands do have the most degrees of freedom of any other parts of our body, and it is this complexity that makes robot hands so difficult to build.

We've all seen the high cost of complex bipedal humanoids and hexapods with 3 DOF legs caused by the large number of servos and complex control circuitry required to create the complex balancing and movements. Each true anthropomorphic robot hand requires just as much complexity and degrees of freedom as these robots, but their smaller size makes it difficult to incorporate servos within the hand's structure. Tiny and expensive micro servos can fit within individual fingers but they lack the strength to approximate human power and digital force. Many robot hand designers use push rods to connect individual finger joints to linear actuators in the wrist and arm structures, but 14 of these take up a lot of room.

Build Your Own Fingers and Hands

One of the simplest methods humanoid robot builders have used to produce realistic human-looking hands is to use some sort of bendable tubing. I once used 3/4 inch Tygon tubing with partial notches at the inside of each joint to facilitate bending at those points. A cable inside the tubing attached at the finger tip and pulled from the other end by a linear actuator caused the finger to bend inward proportionally to how hard and far the cable was pulled. **Figure 3** shows a drawing of the finger mechanism that I made in 1982 before CAD days. Though I had a nice mill and lathe at the time, virtually all of the work in building my robot hands was with hand tools. (No pun intended.)

I fitted each of the fingers and thumb into a very realistic prosthetic glove. It worked pretty well until one of the tubing joints broke through at the V notch. At that point, I wrapped each finger with duct tape to slow down the tearing failure rate, but soon all fingers began to fail. After that, I decided to insert a spring into each finger tube to prevent chafing from the cable. Instead of cutting the slots, I ground notches into the new pieces of tubing with a Dremel tool so that there would be no sharp corners to start a tear. I first tried filling the glove and tubing with a gel but that proved quite messy and it leaked. After cleaning out the gooeey gel, I decided to fill each finger with a soft rubber potting compound and carved part of it out

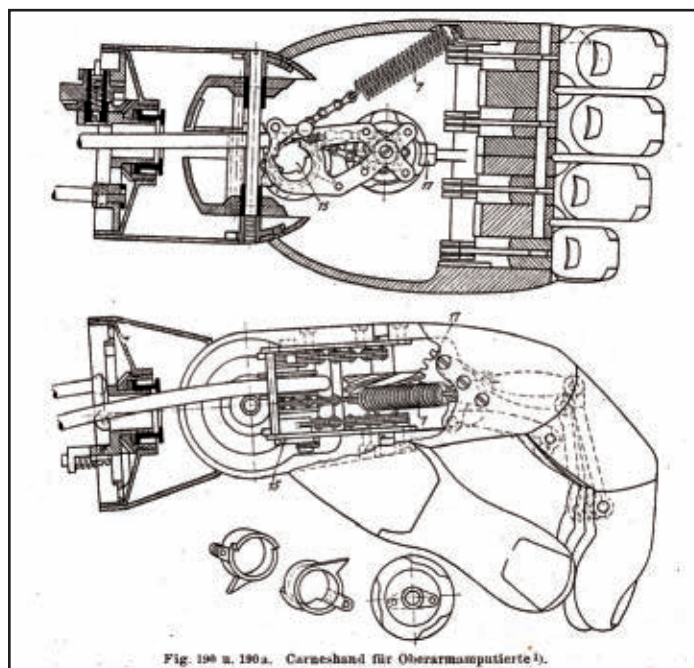
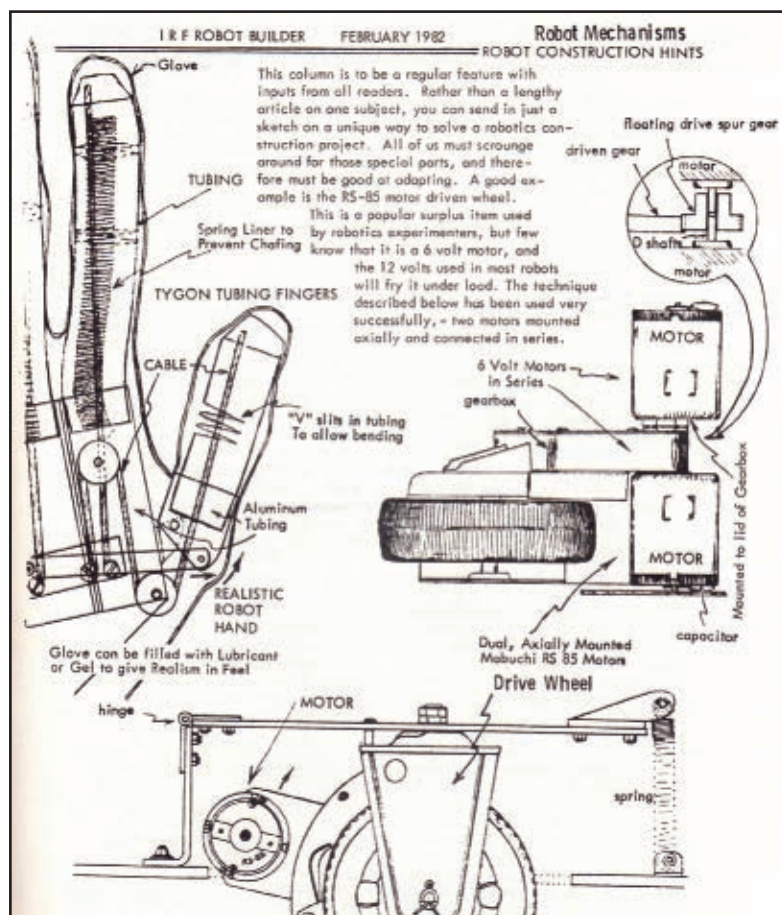


FIGURE 2. Carneshand from 1916 German patent for amputees.

at each V slot. These fingers worked fine but I later set them aside for another finger design project. I have discussed this design with others in years since, and there have been some much better hands built using this method

Robot Hands



FIGURE 4. 'God Hands' from Crafthouse.

with newer and better tubing products.

The best hands and fingers that I ever made used simple 5/8 to 3/4 inch OD 6061 aluminum tubing cut into finger segments and screwed together at the joints. I tried rivets at first but they were hard to keep equal in tightness, and harder to install and remove. Each segment was beveled at the joint to allow inward bending. I made a single cut on the long piece of tubing with a cutting tool with my lathe before I cut it into segments. That gave the fingers a great, shiny look. The same can be done with sandpaper and the tubing chucked in a drill press. I spent days of hand-filing down each of the segments until my hands were black with aluminum dust.

In the assembly process, I used pieces of a straightened clock spring inside the finger segments to hold them straight except when the cable was pulling on the finger segments. I believe the cables that I used were pieces of a bicycle brake cable and the ends of each had a small

FIGURE 6. Complete Melissa robot from Crafthouse.

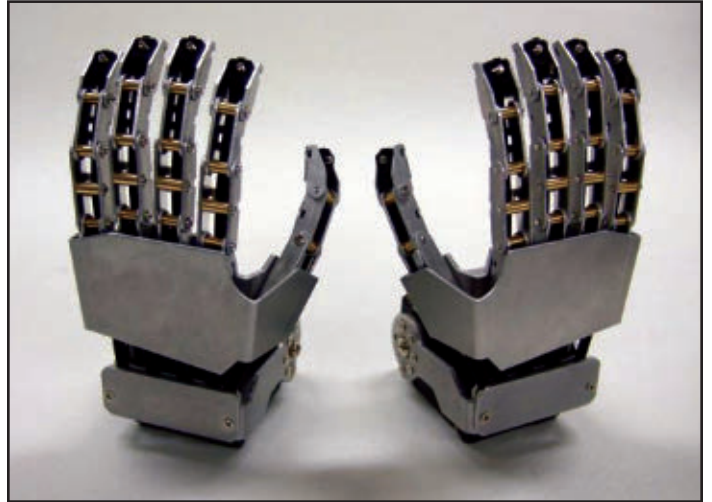


FIGURE 5. Melissa hands by Crafthouse.

tension spring attached to allow each of the four fingers to exhibit individual compliance when grasping an object. I used two small surplus linear actuators: one for the four fingers connected together and the other for the thumb. I wish I still had some photos of these hands but I traded them for some other robot parts back in the late '80s. The first set of hands looked and felt more like human hands within the prosthetic glove, but the second set were way more cool and robotic like — somewhat like the hands I'll discuss next.

Buy Fingers and Hands for Your Robot

If you don't feel adept at crafting a set of hands for your newly-minted humanoid creation, you might want to consider buying a pair from Crafthouse. The original hands seen in **Figure 4** — also called 'God Hands' — are priced at about \$1,935 a pair. If you are inclined to pay your mortgage and eat occasionally in these tough times, the newer set seen in **Figure 5** (called the 'Melissa Hands' Type-



FIGURE 7. Melissa hand spare finger.

1) will set you back only \$645. The Melissa robot is seen in **Figure 6** and looks nothing like any of the Melissa's I know, but you can see by the extended hand and arm that it is of pretty high quality. These hands are a bit larger than the former; suitable for a humanoid biped about 20" tall. You can also buy individual digits (shown in **Figure 7**) for the Melissa hand at \$53 each. There are several Crafterhouse videos on various sites on the Internet that show both of these robots and their hands in action.

Both of the sets of hands come pre-assembled and utilize one Kondo KRS-4014SHV servo that is not included with the hands. The hands were originally designed for use with Kondo KHR series robots but are applicable to most small humanoids. This single servo (that is fairly expensive in its own right) opens and closes all five digits at once — not individually. (It should be noted that potential users should not let their humanoid robot using these hands fall forward to the floor as the hands are fairly delicate and might become smashed.) If you're not designing a robot for any sort of combat or one that might fall and you have money to burn, you might want to consider buying one of David Ng's arms and hand shown in **Figure 8**. The complete 13 degrees of freedom arm/hand goes for about \$4,500, but you can also purchase a single 4 DOF android finger kit for \$55 or assembled for \$110. It's a pretty good bargain considering each finger has 60 parts.

In my opinion, unless you have a lot of money to burn, I'd suggest trying to build your own hands like I talked about earlier. If you are handy with hand tools and can do some precision measurements, you can build hands that look every bit as nice as the ones mentioned here. Though I already had a lot of aluminum sheet and tubing stock in my shop, I think it would have only cost me \$20-\$30 in parts (not including the actuators). If I remember right, I was watching a bunch of college football games on TV on New Year's Day as I filed, sanded, and drilled away. I used a hacksaw and a pair of pliers to cut and bend in the ends at the joints so that they would fit together. It was bend and tweak, try to fit the pieces together, tweak some more, fit, tweak, fit, and so on.

The Festo Bionic Handling Assistant

Figure 9 shows a very unique robot claw with three

FIGURE 9. Festo bionic handling assistant hand.

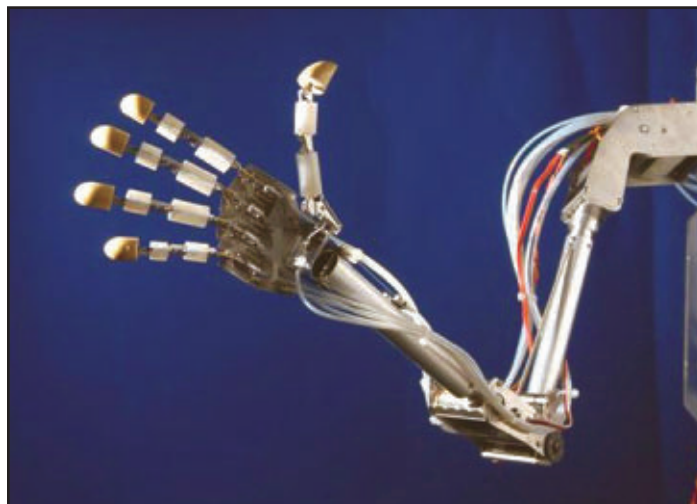


FIGURE 8. The \$4,500 13 DOF hand by David Ng.

fingers. This particular photo is of the Festo 'Bionic Handling Assistant' picking up a foil-wrapped chocolate egg. Notice that the fingers actually curve around the shape of the egg. You might have heard of Festo and their 'Air Penguins' — fully autonomous flying (floating) robots shown in **Figure 10**. Containing about a cubic meter of helium, these creations were a hit in Europe where the German company is located. Festo is a specialist in pneumatics and pneumatic movement. The bionic handling assistant is a pneumatic end-effector attached to a pneumatic arm shown in **Figure 11**. This arm has several sections; each is composed of three long bellows that allow the arm to curve in any direction. The end-effector is also maneuvered by a set of three smaller bellows. (Note the five smaller convolutions behind the finger assembly.)

Final Thoughts

All of these examples are the result of researchers and experimenters taking an idea and just 'going for it.' From my own experience,

FIGURE 10. Festo Air Penguins.





FIGURE 11. Festo bionic handling assistant arm.

I know that many of you can buy just a few dollars of surplus materials and make a set of great robot hands using only parts that you might have around the house. They can be attached to a humanoid that you already have or are building, or they can be used for an arm design that you are developing to assist someone who has lost their arm or hand.

Robot hands seem complex but I assure you that many — if not most — of you can make some every bit as good as the ones covered here. I'd like to challenge you to submit photos and descriptions of your five-fingered humanoid hands and arms, and we'll compile them into an article about reader's projects. **SV**

Tom Carroll can be reached at TWCarroll@aol.com.

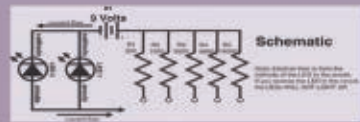


FUNdAMENTALS

For The Beginner

Need the Basics?

Follow along with our series of articles which includes easy to understand graphics. Starting in the May 2010 issue.



Subscribe Today! (with optional digital back issue viewing.)

Visit www.nutsvolts.com or call (800) 783-4624

THE ORIGINAL SINCE 1994

PCB-POOL

Beta LAYOUT

Servicing your complete PCB prototype needs :

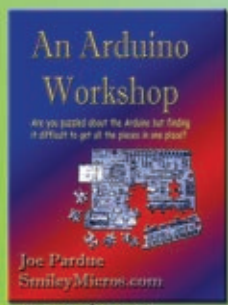
- **Low Cost - High Quality**
PCB Prototypes
- **Easy online Ordering**
- **Full DRC included**
- **Lead-times from 24 hrs**
- **Optional Chemical Tin finish**
no extra cost

FREE LASER STENCIL WITH ALL PROTOTYPE PCB ORDERS

Watch "ur" PCB®
Follow the production of your PCB in **REALTIME**

email : sales@pcb-pool.com
Toll Free USA : 1 877 390 8541
www.pcb-pool.com

Puzzled by the Arduino?



An Arduino Workshop
Are you puzzled about the Arduino but finding it difficult to get all the pieces to see project?
Joe Pardue
SmileyMicros.com
Book \$44.95



Book and Kit Combo
\$124.95

Based on the Nuts&Volts Smileys Workshop, this set gives you all the pieces you need!

For more info on this and other great combos!
Please visit: <http://store.nutsvolts.com>

ROBO-LINKS

GPS MADE SIMPLE™



Linx Technologies.com

LOCATE • TRACK • MAP • FIND • NAVIGATE

THE ORIGINAL SINCE 1994

PCB-POOL®

Beta LAYOUT

- Low Cost PCB prototypes
- Free laser SMT stencil with all Proto orders

WWW.PCB-POOL.COM

RobotShop.com




Pololu
Robotics & Electronics
WWW.POLOLU.COM

ALL ELECTRONICS CORPORATION

Electronic Parts & Supplies Since 1967

AndyMark
Inspiring Mobility



www.andymark.com

ServoCenter servo-center.com

- 16 servos, 16 I/O, 8 A/D
- USB, RS-232, TTL serial
- 14-bit servo control
- Built-in SC-BASIC Sequencer



ServoCenter 4.1 USB \$56.99
ServoCenter 4.1 USB MINI \$59.99

For the finest in robots, parts, and services, go to www.servomagazine.com and click on Robo-Links.

Ultrasonic Ranging is EZ



- High acoustic power, auto calibration, & auto noise handling makes your job easy.
- Robust, compact, industrial (IP67) versions are available.

www.maxbotix.com

Das Blinkenboard

Not just for blinken LEDs.

Much Much More!

Complete kits available @

<http://store.nutsvolts.com> & <http://store.servomagazine.com>



INVEST in your BOT!



INVEST in HiTEC

12115 Paine Street • Poway, CA 92064 • 858-748-6948 • www.hitecrod.com

IMAGES Scientific Instruments

SCIENCE, ROBOTICS & ELECTRONICS

www.imagesco.com

Tele: (718) 966-3694 Fax: (718) 966-3695

Microcontrollers
Servo Control & Motors
Artificial Vision
Speech Recognition

To Advertise: Call 951-371-8497

ADVERTISER INDEX

All Electronics Corp.19, 81	Images, Co. 81	Solarbotics/HVW7
AndyMark48, 81	Linx Technologies81	Technological Arts19
AP Circuits48	Lynxmotion, Inc.13	The Robot Marketplace36
BaneBots7	Maxbotix81	Vantec36
DigilentBack Cover	PCB Pool80, 81	X-treme Geek3
GSS Tech Ed48	Pololu Robotics & Electronics ..17, 81	Yost/ServoCenter81, 83
HiTec2, 81	RobotShop, Inc 81, 82	



Now Serving Europe with Optimized Logistics

- Currency in EURO
- 3 day shipping to 70% of the territory
- Service in French and English
- Competitive shipping rates
- Huge product selection

One Global Door for Manufacturers

VISIT: **www.RobotShop.com** Shipping Worldwide

Robotics at your service! TM

The Future of Servo Control is Calling...

ServoCenter™

Features

- USB, RS232, and TTL serial support
- 16 servos, 16 digital I/O, 8 analog inputs
- Built-in SC-BASIC Sequencer with EEPROM
- Sequencer allows stand-alone operation
- 64 scene presets stored in EEPROM
- Presets instantly loaded or cross-faded
- Built-in configurable smoothing algorithm
- Independent, simultaneous speed & position control
- Scaled, percentage, and group movement commands
- Timed movement commands
- Max, min, & startup position settings
- Ultra-precise 0.05425µS jitter-free pulse resolution



\$56.99

ServoCenter 4.1 USB

Flexibility

- User upgradeable firmware
- Upload your own firmware with bootloader
- Watchdog timer for failsafe operation
- Over-current, over-temperature, polarity protection
- Internal regulator, external power, or battery power options
- Supports 4.8/6.0V regulated servo supply voltages at 5 Amps
- Each digital I/O and analog input has supply pins
- Direct serial, activeX control, or Win32 DLL communication
- Programming examples in 10+ languages
- Windows, Linux, Mac OSX compatible

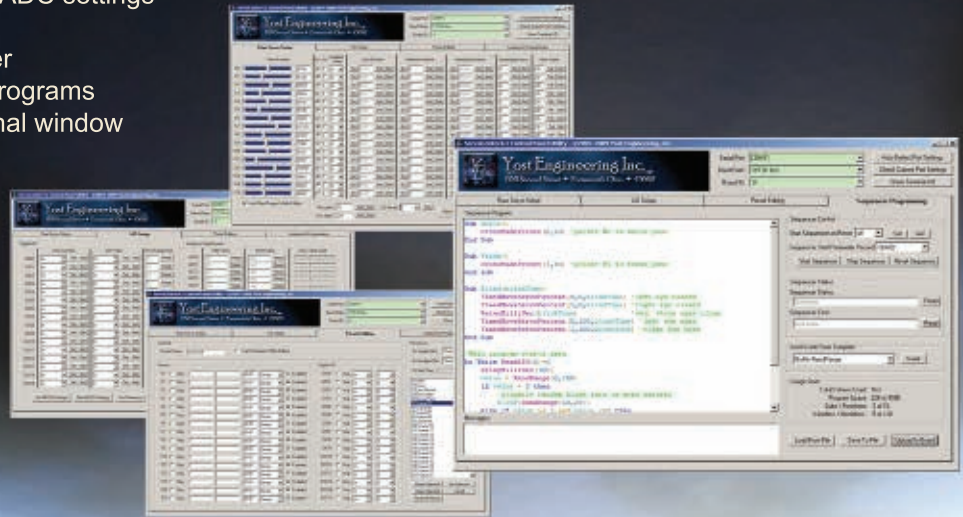


\$59.99

ServoCenter 4.1
USB Mini

Free Control Panel Software

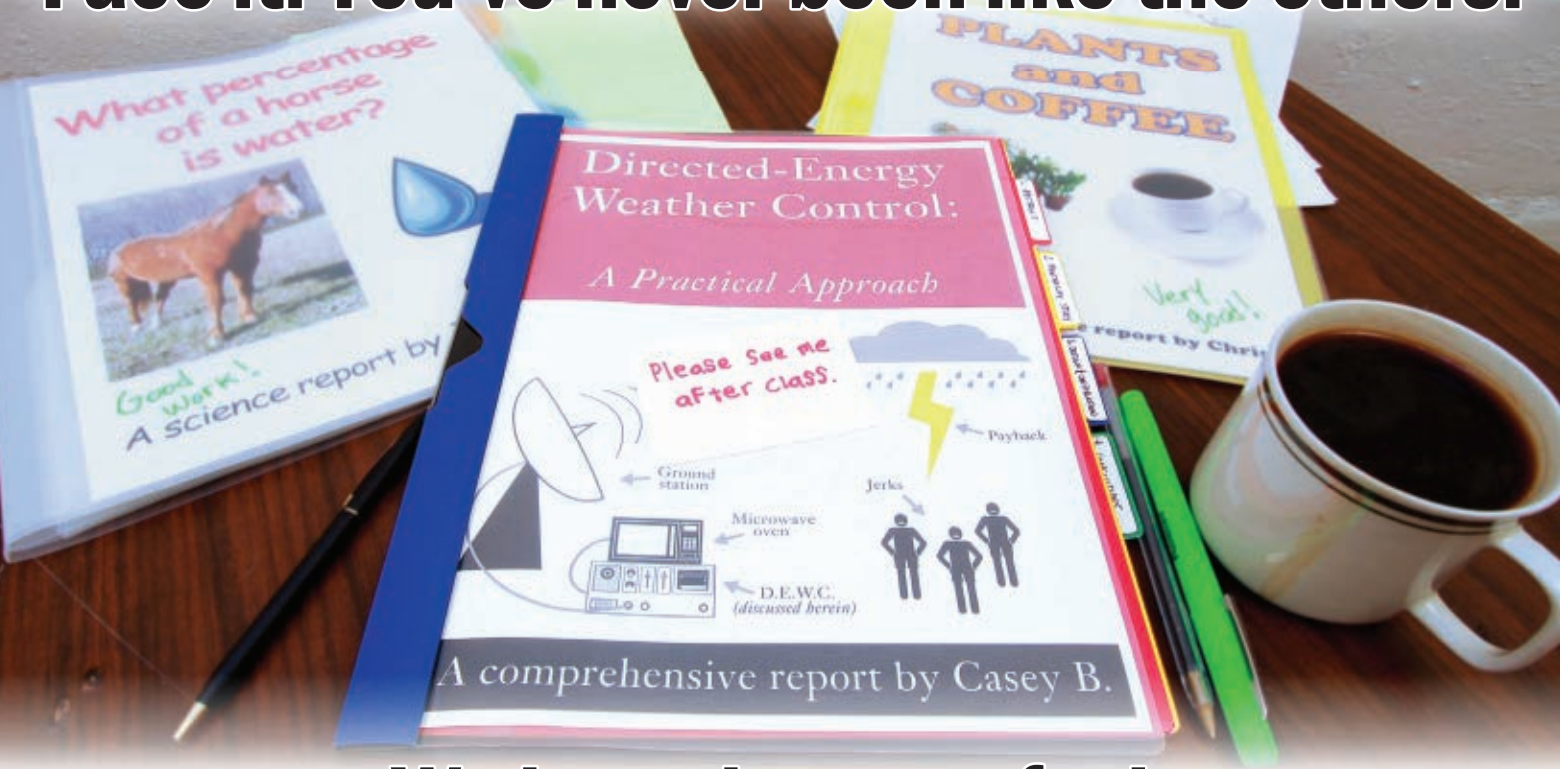
- Easy editing and configuration of servo settings
- Configure and set up digital I/O & ADC settings
- Edit scene presets
- Program the SC-BASIC sequencer
- Upload and run your SC-BASIC programs
- Debug & communicate with terminal window



www.Servo-Center.com

Yost Engineering Inc.

Face it. You've never been like the others.



We know how you feel.

CEREBOT™
32MX4

\$74

Enter the value code "servo2011a" on our website for this special introductory price

A powerful PIC32® based microcontroller board, ideal for robotic experimentation in C, C++, and Assembler.

- Microchip® PIC32MX460F512L
- 80 MHz 32-bit MIPS processor
- 512K Flash, 32K RAM
- Eight R/C servo connectors, two SPI ports, two I²C ports, two UARTs
- Nine 12-pin Pmod™ Connectors
- Five 16-bit timers with 5 input capture and 5 PWM outputs
- 16 channel, 10-bit, 500ksps A/D converter
- On-board USB Program / Debug Interface
- USB OTG Host / Device Capable
- USB Powered



Pmod™
Peripheral Modules

Expand your designs
with sensors, actuators, and other I/O.

\$9.99 - \$29.99



Pmod ENG
Rotary encoder module
w/ integral pushbutton



Pmod HBS
2A H-bridge
w/ feedback inputs



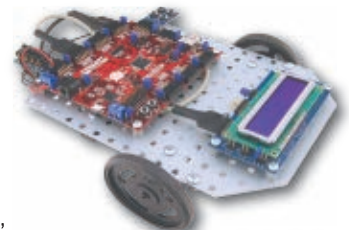
Pmod JSTK
2-axis joystick
w/ 3 pushbuttons



Pmod CLS
2-line character LCD
w/ serial interface

RDK
ROBOTIC
DEVELOPMENT
KITS

Complete kits include the Cerebot™ 32MX4, chassis, gearmotors, wheels, sensors, and software.



Starting at **\$149**

**ELECTRONICS
EXPLORER™**



Analog design revealed

A complete, circuit development kit that includes a breadboard, programmable power supplies, and powerful test instruments.

- 4-channel 40 MSa oscilloscope
- 2-channel arbitrary waveform generator
- 32-channel logic analyzer
- 32-channel digital pattern generator
- ...and more.

\$399 (Academic)

Complete System!
Includes EE Board + Software

DIGILENT®
www.digilent.com